A COM-BASED GRAPHICAL USER INTERFACE FOR A DECISION SUPPORT SOFTWARE

By

YU AN

Bachelor of Science

Sichuan University

Chengdu, China

1995

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2000

A COM-BASED GRAPHICAL USER INTERFACE

FOR A DECISION SUPPORT SOFTWARE

Thesis Approved:

[illegible]

Thesis Adviser

[illegible]

[illegible]

Wayne B. Powell

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my thesis advisor, Dr. K. M. George for his valuable advice, encouragement, and guidance through my M. S. study. Sincere gratitude also goes to my thesis committee members, Dr. Hedrick, and Dr. Dai. This thesis would not be possible without their assistance and encouragement.

I wish to thank to Department of Computer Sciences for awarding me the Phillips scholarship and offering me the chance to pursue my Master degree under direction of many kind and knowledgeable faculty.

I also want to thank my husband, Kun Cheng, for his love, support, patience, and understanding. To my parents, Wu Chaozheng and Ding Zhengguo, who came all the way from China to take care of my life and show their love and support.

Last , but not least, thanks to all my fellow students and friends.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Decision support systems are software used to abstract system behavior and to help decision-makers to make high quality decisions. Principal component analysis (PCA) can be used in the decision support system to reduce the dimension of a data set that consists of several interrelated variables, while retaining the information presented in the data set as much as possible [19].

Wen developed a decision support system based on PCA [19]. It loads data from a file, ranks items based on the data using PCA, and provides methods for performing sensitivity analysis on the ranking. Also it provides a simple Graphical User Interface (GUI) to view the results.

The decision support system developed in [19] lacks in several areas of user interface. A user would like to have a system whose interface resembles the modern MS Windows based interface with its associated features. It would be useful to provide capabilities to exchange data between this application and other standard or custom applications. The objective of this thesis is to design and develop a user-friendly interface with a more extensive set of features based on the **c**omponent **o**bject **m**odel (COM) for the decision support system.

The new system is more adaptive to different input and output sources. On the input side, text files, relational database tables and Microsoft Excel workbook files are acceptable. On the output side, both text files and CSV (Comma delimited) files are available. The new system is also more convenient to use. The toolbars have been customized to share functionality with some menus. Each view's contents can be exported to Microsoft Excel directly.

Due to this design and implementation approach being based on COM, the system should be easier to develop and maintain in the future. This COM model divides the whole system into a two-tier project. The front-end user interface takes care of input collection and output presentation. The data calculation process between input and output is wrapped into a COM object acting as back-end server. The user interface and the server object communicate with each other via a component interface that provides inter-process function calls [3]. One of the characteristics of this model is that tasks are separated into different components. Consequently any changes made to a particular component have no affect on other components. Thus, it makes the system more flexible to develop and maintain.

The remainder of the thesis consists of four chapters. Chapter 2 reviews past research related to user interface design and implementation and Component Object Model (COM). Chapter 3, Design and Implementation focuses on three main development areas of the project, COM-based architecture, diverse input and output data sources, and data exchange with other applications. Results and discussion are shown in Chapter4. Chapter 5 concludes the project and proposes future work on it.

CHAPTER II

Literature Review

In this chapter we review issues related to user interface design and implementation. The Microsoft Visual C++ environment for user interface and Component Object Model implementation is also reviewed.

2.1 User Interface Overview

The human-computer interface is that part of a software system that allows a user to enter, store, manipulate and retrieve data, and initiate commands [10]. It enables communication between the user and the computer. The interface is an important factor contributing to the usability and to the first impression of a system.

The user interfaces have evolved over the years from command line data entry to the current generation of GUIs. On an IBM compatible personal computer (PC), the DOS interface is an example of a command line interface. A command line interface gives the user more opportunity to initiate and control the dialogue.

Better interface was considered to be a requirement to achieve the increase in productivity that was being promised through the use of computers. This awareness led to advances that made the interface more visual using techniques such as menus to ease the burden on users' memories [15]. This style of interface, called a menu driven interface, can be developed into a hierarchical menu system, where the main menu leads to sub-menus that themselves may lead to sub-sub-menus, continuing down an indeterminate number of levels before the functionality of the program is available to the user. Although menu driven interfaces are easy for the developer to construct they are very restrictive to the user as the interface designer has to decide at the outset which options are to be accessed from which menus and in what sequence the options are to be presented [17].

Interface technology has been evolving continuously. The current point and click GUI (Graphical User Interface) is adopted universally and is used in most applications. Applications typically used the elements of the GUI that came with the operating system then added their own graphical user interface elements and ideas to make using a computer easier. Today the GUI familiar to most of us in either the Mac or the Windows operating systems and their applications originated at the Xerox Palo Alto Research Laboratory in the late 1970s. Apple used it in their first Macintosh computers. Later, Microsoft used many of the same ideas in their first version of the Windows operating system for IBM-compatible PCs. currently the most popular user interface is possibly the **w**indows, **i**cons, **m**enus and **p**ointer (***WIMPS***) style. This style of interface is used with Microsoft Windows, Apple computers, and X Windows, and has become very acceptable to wide variety of users. This thesis develops and implements a MS Windows based user interface to a decision support system software [19].

2.2. Graphical User Interface (GUI)

As mentioned in the previous section, the graphical user interface we experience today originated in late 1970's at the Xerox Palo Alto research laboratory and utilizes computer graphics capabilities to make the applications easier to use. The primary interaction mechanism is some kind of pointing device that is the electronic equivalent to human hand. Commands and actions are encapsulated within icons, thus making it easy to comprehend and use. These icons serve as the primary vehicle of interaction between applications and users [2]. Basically there are two styles of interaction for graphical systems, "direct manipulation" and "indirect manipulation".

As early as 1982 Shneiderman used the term "direct manipulation" to describe a style of interface that is designed as an extension of the real world [15]. In this system graphical elements replicate real world objects and provide similar continuous visibility. However, direct manipulation might not be applicable in practice under certain circumstances. For example, a window's space might be too limited to provide enough space for all the objects. Typing sometimes may replace pointing. Nowadays, most window systems are a mix of both direct and indirect manipulation. The interfaces developed in this thesis can be classified as belonging to this group.

2.2.1. Characteristics of GUI

Compared to command line and event-driven systems, a GUI system has some characteristics that make it widely accepted and applied to increasingly many applications.

1. Sophisticated visual presentation

   Most GUI systems allow display of information in a variety of formats – drawings, color text, video, etc. Also, several controls are available to manage the presentation of information.

2. Object Orientation

   Current GUIs are designed and developed using the object-oriented programming model. The benefits of the model are ease of development, information sharing, and extension. MS Windows 95 (and later versions) belongs to this class of interfaces. In Windows 95 there are several kinds of relationships called collections, constraints, composites, and containers [8].

3. Limited control options.

   The user can manipulate only the currently visible window (i.e. the window that has the focus) and no other.

4. Simple interaction

   The user depends on a simple pointing device, a mouse in most cases, to interact with a graphical system. A complete action can be accomplished by point and click.

5. Multitasking

   It's very common that graphical systems process more than one task at the same time. The system can switch between multiple active tasks to maximize efficiency [20].

6. Easy transfer of data

   In the Microsoft windows system it is convenient to use the clipboard or drag and drop function to move objects between different applications.

2.2.2. Interface Design

Designing a good user interface is always a complex task. It should be useful and reflect user's capabilities and respond to his or her specific needs. Better user interface design can achieve the very real benefits in increased productivity, decreased training time, decreased user errors, decreased development time, decreased customer support cost and increased sale that better user interfaces can produce. Poor design is an expensive mistake. Because if the important features are not easily available to users, no matter how innovative or creative the technical solutions might be, they will be wasted [5, 7].

General principals

1. Compatibility
   - User compatibility

     "Know the user" is the fundamental principle, from which all others derive. A very common assumption of developers is that all users are all alike. This error leads to conclusion that if an interface is fine to a few users, it will be good to all others. Another common assumption is that users feel and think like developers. It has been proved that users usually differ significantly from developers in motivation, sophistication, and needs [23]. As a good interface designer, must know and understand different aspects of users and adapt the interface to different needs.

- Task compatibility

  The structure and flow of a system should match the task that user must do to perform the job. The organization of functions should allow easy transition between tasks. Therefore task-oriented style should be preferred to application-oriented style [15].

- Product compatibility

  More often than usual, users are already familiar with other graphical system or an earlier version of the new system. Designers should take their habits and expectation into account when designing a new system. Otherwise, users must start over to learn the new system from beginning. Compatibility across system can reduce learning time and errors significantly [15].

However, compatibility shouldn't be overemphasized to the exclusion of technical improvement over an older system, but it obviously should be one of the concerns of developers in making design decisions.

2. Consistency

   Consistency is very important because it permits people to reason by analogy and predict how to perform jobs even if they've never encountered them before [8]. Inconsistency in most cases increases learning requirements. Excess learning requirements are a barrier to people's achieving high performance, and they possibly influence their acceptance of the system [13]. General design standards or guidelines can help to maintain design consistency. Since late 1980s GUI industry and research organizations have developed many guideline documents. These documents specify how different controls of user interface behave and what they look like. They also help designers decide when to use various elements.

3. Simplicity

   A complex interface is overwhelming and confusing to users. Unfortunately, many business applications are so big and complicated that designers find it very hard to provide corresponding complex functionality through a simple interface [1]. There are a few methods that could help this situation.

   - Layered approach

     Present fundamentals to new users on first encounter and gradually introduce advanced functions on deeper layers. The complexity in this way can be distributed in order of layers and it makes novices comfortable while introducing system components step by step [17].

   - Defaults

     Defaults like the layered approach hide some advanced functions from novices so that they don't need to be burdened to make all decisions and can concentrate on fundamentals first [17].

   - Simple common tasks

     Studies have proved that users often need to perform certain set of common tasks. Making common tasks simple increases the whole system's efficiency [17].

4. Control

   People prefer to feel control over the system when they are interacting with a computer application. A tool-like interface is a good choice in that people find useful and easy to master. People easily can gain the feeling of control if a simple, consistent, and flexible interface is presented [23].

5. Flexibility

   Flexibility reflects system's capability to respond to users needs [14]. Different people have different needs based on their knowledge, experience, and preference. A well-designed interface accommodates these needs by providing variety of ways to access function and perform tasks. Interfaces that allow user tailoring and user selections provide more flexibility.

Design Process

The design process used in this thesis is as follows [15]:

1. Define problem domain
2. Analyze tasks
3. Define corresponding objects and functions
4. Design GUI's appearance and behavior
5. Evaluation

1. Define problem domain

This is the most important and basic step of design process. Even a small error during this step can lead to major changes and corrections in subsequent steps. It is important that domain concepts be identified as explicitly as possible. Multiple tools, such as UML (Unified Model Language) products are available to help developers collect business requirements and analyze use cases.

2. Analyze tasks

This step is to help developers understand what the users want to accomplish, the strategies the users use in meeting their goals [18]. This information is essential to designing a task-oriented system. Based on previous steps, this is accomplished by determining basic business functions, describing user activities through task analysis, understanding the user's mental model, and developing a conceptual model of the system.

3. Define corresponding objects and functions

In this step designers determine the graphic objects to be selected for the user interface, and the functionality to be accomplished by these objects. Also, specific methods of communication between objects must be defined.

4. Design GUI's appearance and behavior

The purpose of this stage of design is to define a user interface in a way that is attractive and compatible with user's habits and expectations.

5. Evaluation

The real test of an interface is the behavior in practice, but it is costly to modify the interface at the stage of implementation. Thus it is important to evaluate the interface prior to its actual implementation. Gould suggested different technique to be used in evaluation [15].

1. Printed or video scenarios indicate the concepts captured in interface
2. Mock-ups demonstrate the appearance of the user interface
3. Formal prototype test provides feedback on how well sample users can perform specific tasks.

The user interface design process is iterative. Evaluation should be performed at each step. When problems are found, corrective steps should be taken in the phase in which deficiencies were found.

2.3. Decision Support System

As mentioned in the introduction, decision support systems are software used to abstract system behavior and help decision makers to make high quality decisions from a large interrelated data information system. Traditional methods including simulation and modeling have been used widely to address the problem. In [6], George presented this kind of decision support system to select a choice from many alternatives.

Karl Pearson was the first researcher to develop principal component analysis (PCA) [9]. Since then other researchers have done comprehensive work in this area. An important step in interpreting the data is the reduction of the dimension. Simplification of data is another step. Principal component analysis can address these issues, while keeping most of the information present in the data set. It is widely used in many areas including agriculture, biology, chemistry, geology, and quality control. This thesis is based on the work done by Wen to develop a decision support system using PCA and presented in [19].

2. 4. Implementation Tools and Environment

There are currently many implementation tools available on the market to implement software systems, especially user interfaces. Sun's JDK (Java Development Kit) provides a cheap way to develop Java applets that are appealing to Web developers. Apple's OpenGL SDK (Standard Development Kit) is a standard 3D graphics API (Application Programming Interface) for Power PC based interface. Microsoft Windows provides several tools for developing GUIs including the visual programming languages MS Visual Basic and MS Visual C++ that are considered popular choices for developers. Due to the dominance of windows application on PC, Microsoft GUI tools are widely accepted for PC interface implementation [2]. Visual C++ is used to develop the software in this research. Therefore, a brief description of the available tools is given in the following sections.

2.4.1. Microsoft Visual C++ environment

Visual C++ provides an Integrated Development Environment (IDE) that allows developers to write, build, and debug a 32-bit user interface program [21]. Also, Visual C++ gives developers a set of application wizards that guide the developer though creating a new program. An application wizard is a tool in Microsoft Visual Studio that can generate a new, fully functional application from which developers build a more complex application. Included in the AppWizards are basic settings and structure code. It saves considerable time especially for experienced developers [3]. A typical single document Windows interface can be created using the MFC (Microsoft Foundation Class library) AppWizard (exe). Developers are free to select basic structures and options, such as Document/View architecture, database support, and Automation.

Microsoft Foundation Class library is the core of MS Visual C++ [21]. Basically MFC consists of a large number of classes. Most of these classes are derived from a single root class at the top of the class hierarchy. CObject, with very low overhead, supports serializing data and obtaining run-time class information. CButton class that encapsulates all the information for Windows button control can serve as a good example of the class hierarchy.

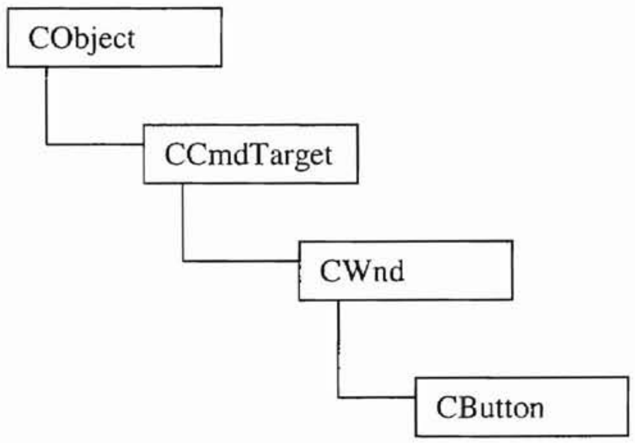


Figure 1. Cbutton in Class hierarchy

The most frequently used classes for implementing user interface are:

Frame Window Classes (windows)

View Classes (windows)

Dialog Box Classes

Control Classes

Control Bar Classes

Frame window and View are part of the basic structure of a Windows user interface. Control and Control Bar are attached to the basic structure to present event-driven functionality behind them. Dialog is a special kind of window that performs more complicated tasks than regular controls do like search-and-replace operation. Besides these classes directly related to user interfaces, MFC also provides file and database classes, drawing and printing classes, and networking classes. They are listed below along with short descriptions of their use.

Cstring methods:

int GetLength( ) const return the count of the bytes in this **CString** object not including the null terminator

Bool Cstring::IsEmpty Tests whether the object contains no characters

Void Cstring::Empty Force the object to be empty and reclaim the memory

int Compare(LPCTSTR *lpsz*) const compare two strings (case sensitive) return zero if identical; < 0 if this **CString** object is less than *lpsz*, or > 0 if this **CString** object is greater than *lpsz*.

int CompareNoCase(LPCTSTR *lpsz*) const Compares two strings (case insensitive) return zero if identical; < 0 if this **CString** object is less than *lpsz*, or > 0 if this **CString** object is greater than *lpsz*.

void Format(LPCTSTR *lpszFormat*, ...)   write formatted data to a **CString** in the same way that **sprintf** formats data into a C-style character array.

CString Left(int *nCount*) const   Extract and return the left part of the string (up to nCount length)
throw(CMemoryException)

CString Right(int *nCount*) const   Extract and return the right part of the string (up to nCount length)
throw(CmemoryException)

void TrimLeft( )   Trim the leading whitespace characters from the string

void TrimRight( )   Trim the trailing whitespace characters from the string

int Find(LPCTSTR *lpszSub*) const   find the substring (lpszSub) inside the large string

CString SpanExcluding(LPCTSTR *lpszCharSet*) const throw(CMemoryException)
Extract and return a substring that contains only the characters not in a set(lpszCharSet)

Cdialog methods:

virtual int DoModal( )Calls a modal dialog box and returns when done. The return value could be IDOK, IDCANCEL, or IDABORT.

Cdialogbar methods:
int GetDlgItemText(int *nID*, CString& *rString*) const   Retrieves the caption or text associated with a control. Return the actual number of bytes copied to the buffer, not including the terminating null character. The value is 0 if no text is copied

void SetDlgItemText(int *nID*, LPCTSTR *lpszString*)   Sets the caption or text of a control in the specified dialog box

CWnd methods:

CWnd* GetDlgItem(int *nID*) const Retrieves the control with the specified ID from the specified dialog box.

void SetWindowText(LPCTSTR *lpszString*)   Sets the window text or caption title (if it has one) to the specified text.

void GetWindowText(CString& *rString*) const Returns the window text or caption title (if it has one).

BOOL ShowWindow(int *nCmdShow*) Sets the visibility state of the window. Return Nonzero if the window was previously visible; 0 if the **CWnd** was previously hidden

CMenu* GetMenu( ) const Retrieves a pointer to the menu for this window.

CrichEditView methods:

CRichEditCtrl& GetRichEditCtrl( ) const Retrieves the rich edit control.

CrichEditCtrl methods:

int GetLineCount( ) const Retrieves the number of lines in the object
int GetLine(int *nIndex*, LPTSTR *lpszBuffer*) const Retrieves a line of text from the object

DWORD GetDefaultCharFormat(CHARFORMAT& *cf*) const Retrieves the current default character formatting attributes in the object

BOOL SetDefaultCharFormat(CHARFORMAT& *cf*) Sets the current default character formatting attributes in the object.

void SetSel(long *nStartChar*, long *nEndChar*) Sets the selection in the object

void ReplaceSel(LPCTSTR *lpszNewText*, BOOL *bCanUndo* = FALSE) Replaces the current selection in the object with specified text.

void Clear( ) Clears the current selection

Cmenu methods:

BOOL LoadMenu(UINT *nIDResource*) loads a menu resource from the application's executable file and attach it to the **CMenu** object. Returns nonzero if the menu resource was loaded successfully; otherwise returns 0.

CMenu* GetSubMenu(int *nPos*) const Retrieves the **CMenu** object of a pop-up menu.

BOOL TrackPopupMenu (UINT *nFlags*, int *x*, int *y*, CWnd* *pWnd*, LPCRECT *lpRect* = 0) Displays a floating pop-up menu at the specified location and tracks the selection of items on the pop-up menu.

Cfile methods:

virtual BOOL Open(LPCTSTR *lpszFileName*, UINT *nOpenFlags*, CFileException* *pError* = NULL) Safely opens a file with an error-testing option

virtual void Close( ); throw(CFileException) Closes a file and deletes the object

virtual void Write(const void* *lpBuf*, UINT *nCount*); throw(CFileException) Writes (unbuffered) data in a file to the current file position.

CDatabase methods:

virtual BOOL Open(LPCTSTR *lpszDSN*, BOOL *bExclusive* = FALSE, BOOL *bReadOnly* = FALSE, LPCTSTR *lpszConnect* = "ODBC;", BOOL *bUseCursorLib* = TRUE)
throw(CDBException, CMemoryException) Establishes a connection to a data source (through an ODBC driver). Returns nonzero if the connection is successfully made; otherwise returns 0 if the user chooses cancel when presented a dialog box asking for more connection information.

virtual void Close( ) Closes the data source connection.

CdaoDatabase methods:

void GetTableDefInfo(int *nIndex*, CDaoTableDefInfo& *tabledefinfo*, DWORD *dwInfoOptions* = AFX_DAO_PRIMARY_INFO)
throw(CDaoException, CMemoryException) Returns information about a specified table in the database.

short GetTableDefCount( ) throw(CDaoException, CMemoryException) Returns the number of tables defined in the database.

CrecordSet methods:

virtual BOOL Open(UINT *nOpenType* = AFX_DB_USE_DEFAULT_TYPE, LPCTSTR *lpszSQL* = NULL, DWORD *dwOptions* = none)
throw(CDBException, CMemoryException) Opens the recordset by retrieving the table or performing the query that the recordset represents. Returns nonzero if the

CRecordset object was successfully opened; otherwise returns 0 if CDatabase::Open (if called) returns 0.

virtual void Close( )   Closes the recordset and the ODBC HSTMT(statement handle) associated with it.

void MoveLast( )
throw(CDBException, CMemoryException)   Positions the current record on the last record or on the last rowset.

BOOL IsBOF( ) const   Returns nonzero if the recordset has been positioned before the first record. There is no current record.

BOOL IsEOF( ) const   Returns nonzero if the recordset has been positioned after the last record. There is no current record.

void SetAbsolutePosition(long *nRows*)
throw(CDBException, CMemoryException)   Positions the recordset on the record corresponding to the specified record number.

void GetFieldValue(short *nIndex*, CString& *strValue*)
throw(CDBException, CMemoryException)   Returns the value of a field in a recordset.

CdaoRecordSet methods:

void GetFieldInfo(int *nIndex*, CDaoFieldInfo& *fieldinfo*, DWORD *dwInfoOptions* = AFX_DAO_PRIMARY_INFO);
throw(CDaoException, CMemoryException)   Returns specific kinds of information about the fields in the recordset.

short GetFieldCount( ) throw(CDaoException, CMemoryException)   Returns a value that represents the number of fields in a recordset.

2.4.2. Component Object Model (COM)

COM refers to both specification and implementation model developed by Microsoft [12]. It provides a framework for integrating component software. This framework supports interoperability and reusability of distributed objects by allowing developers to build systems by assembling reusable components from different vendors that communicate via COM. By applying COM to build systems from existing components, developers are able to enhance the maintainability and adaptability of the system.

COM defines an Application Programming Interface (API) to allow creation of components for use in integrating custom applications or to allow different components to interact as long as they adhere to a binary structure specified by Microsoft [3, 12]. Even the components written in different languages can communicate with one another.

COM makes it possible to divide a whole system into multiple tiers and distribute individual components in different processes or even different computers connected by a network. As shown in Figure 2, the client that is most often a user interface can interact with these components (objects) via interface pointer. Figure 3 provides a more detailed insight into the interaction. The client maintains a pointer to the interface that is a pointer to a pointer to an array of pointers to the object's implementation of the interface member functions. The pointer to the interface function table by convention is called pVtbl pointer. The interface function table is called vtble. It is independent of programming language or tool used to develop it. Therefore, this structure provides the basis for interoperability between software components written in different language [3]. Particularly, Microsoft Interface Definition Language (IDL) is used to define the interface method prototypes. The IDL is an extension of the DCE (Distributed Computing Environment) interface language standard. A COM object can support any number of interfaces. An interface provides a grouped collection of related methods. By calling these methods, a client application feeds input to objects and receives the output from them.

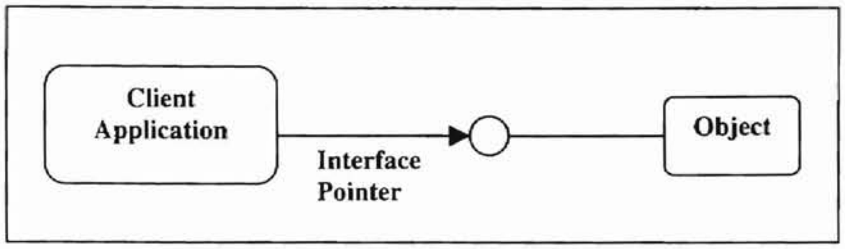


Figure 2. Client to object interface

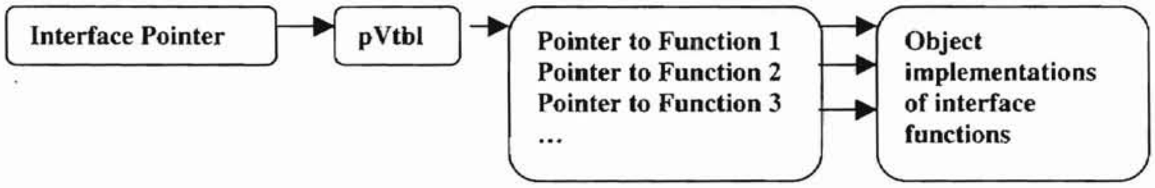


Figure 3. The interface structure

CHAPTER III

Design and Implementation of User Interface

In order to present a user-friendly interface for application of principal component analysis to a decision support system, a window based GUI has been designed in this thesis. We develop an implementation model based on COM to represent the problem and provide an implementation. Written in Visual C++ 6.0, the interface is a typical event driven model. Based on a framework structure provided by MFC Class Wizard, it is customized to bind selectable menus and toolbars to functionality. As a result, users are able to specify input data from different storage such as file systems or relational databases and get results in desired formats.

All the functionality are available through six menus that include four standard menus, File, Edit, View, and Help. The other two additional menus are Data and Sensitivity Analysis. Under Data menu there are four menu items for data calculations. They are Calculate Correlation (CC), Estimate Eigenvector (EE), Estimate Rank (ER), and Sorted Result (SR). Under Sensitivity Analysis menu there are two menu items, Interval for Rank, and Interval for Weight. These six menu items under Data and Sensitivity Analysis menus are also available through toolbars. All the documents visible in client area are printable and compatible to standard clipboard (Cut, Copy, Past, Drag & Drop). The system supports both pure text(local or remote site) and relational database input sources although data must be in the required format. On the other hand, output could be pure text or csv, which is comma-delimited.

3.1. System Architecture

The system presented in this thesis is a COM-based system as described earlier. It supports interoperability and reusability of distributed objects by allowing developers to build system by assembling components designed and implemented by different groups. In this way it enhances the maintainability and adaptability of the system.

The work performed as part of this thesis builds a COM component in Microsoft Visual C++ 6.0 environment. Particularly, ATL (Active Type Library) provides a framework for the component. The software architecture of the system is shown in figure 4.

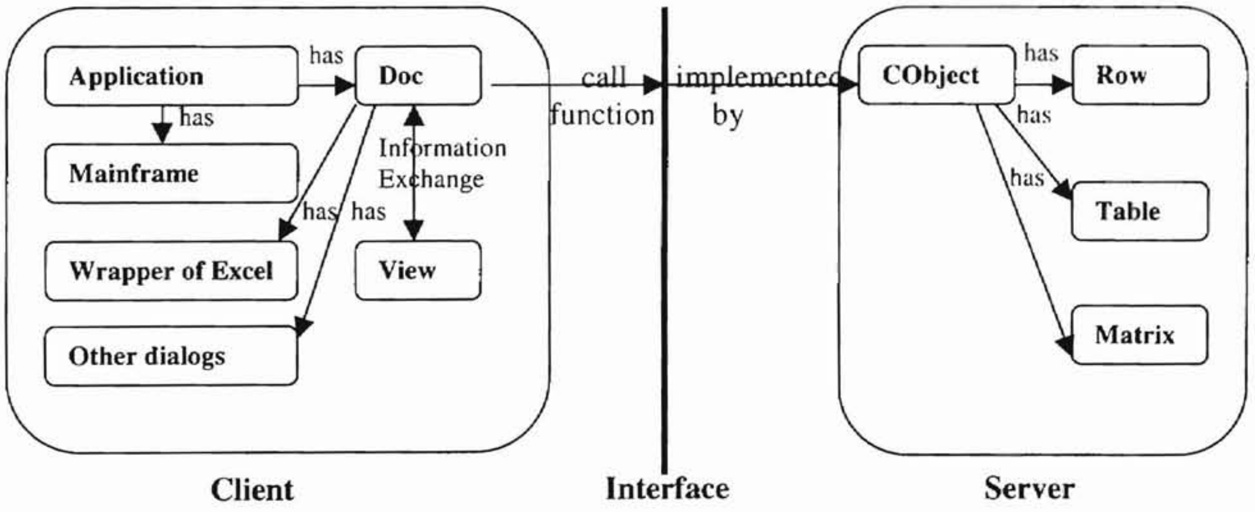


Figure 4. Architecture of COM-based GUI system

On the left side of interface is the GUI that is a typical Windows application of document/view architecture. The main GUI application contains a document and a mainframe structure. The document exchanges document information with view and interacts with other classes at the same time. On the right side of interface is the COM component that implements the interface. Calculation logic consisting of Row, Table, and Matrix classes are hidden from the interface and GUI.

3.1.1. IDL Interface

Client side GUI and server side component communicate via COM interface. The interface is written in IDL as we mentioned earlier. This IDL file (server.idl) is the most important part of the whole system in that it defines what functionality the COM component exposes and how to interact with the component. Any client that desires to interact with the component must have a copy of the IDL file before establishing real connection. Code below shows how IDL defines a COM object. First part describes properties of the component. Second one declares interface functions.

```
import "oaidl.idl";
import "ocidl.idl";

// These are attributes of the interface object

        [
                object,
                uuid(A21118E2-55AD-41AF-A299-42CE1A4F71BC),
                dual,
                helpstring("Iobject1 Interface"),       // The description of the interface
                pointer_default(unique)
        ]
```

```
// The declarations of methods in this interface

        interface Iobject1 : IDispatch
        {
                [id(1), helpstring("method NewTable")] HRESULT NewTable([in] int
                        row, [in] int col, [in] double init);
                [id(2), helpstring("method testing")] HRESULT testing([out, retval] int*
                        col);
                [id(3), helpstring("method WriteRow")] HRESULT WriteRow([in] double
                        *temp_row, [in] int i);
                [id(4), helpstring("method GetRowValues")] HRESULT
                        GetRowValues([in] int i, [out, retval] double *temp_row);
                        .
                        .
                        .
        };
```

The IDL file contains the GUIDs (Globally Unique Identifier) of object and interface that consist of 128 bits. The GUID number is created by running GUIDgen.exe which defines a GUID unique over both space and time. To be unique in space, each GUID requires a 48-bit value unique to the computer on which it is generated. Use of this value guarantees that a GUID generated by any individual computer is different from the GUID generated by another computer. The other 60 bits of GUID stands for time stamp. It represents the count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582. The concept of GUID originated from Open Software Foundation (OSF) [4].

Besides GUIDs of object and interface, the IDL file also contains entries of interface functions that are implemented by the COM object. Each entry of function consists of an id, help string (description of the function), and the prototypes of the function. The return value of all interface functions must be of type HRESULT for the purpose of detecting possible network failure. The *in* and *out* parameter attributes indicate the flow directions of the parameters. It is from client to component (*in*) or from component to client (*out*).

The implementation of these functions is done in server object (COM component). The implementation for one of the functions defined in the above IDL file (WriteRow) is given below.

```
HRESULT Cobject::WriteRow(double *temp_row, int i)
{
        // system generated
        AFX_MANAGE_STATE(AfxGetStaticModuleState())

        int rowNum, colNum;
        rowNum = data->GetRow();
        colNum = data->GetCol();

        // copy a row of input data to the corresponding row of the Table data.
        (*data)[i].input(temp_row);

        // successfully complete the method call
        return S_OK;
}
```

3.1.2. COM Communication Establishment

In order for the client to establish communication with COM component it is necessary at first to register the object and interface in Windows registry. So a running client can look them up through the registry. A type library (server.tlb) which is actually the binary version of IDL file is loaded and registered in the operating system registry. Following code shows how to get the type library from current directory where the executable GUI resides and then load and register it in Windows registry.

```
        OLECHAR             wPath[256];
        ITypeLib*           pTypeLib;

        // load server's type library under current directory
        GetCurrentDirectory((DWORD)256, wPath);
        HRESULT             hr = LoadTypeLibEx(wPath, REGKIND_REGISTER,
                            &pTypeLib);

        // There is an error to load the type library
```

```
if (FAILED(hr))
{
        AfxMessageBox("failed to load type library");
        return false;
}

// successfully load the type library. Release it afterwards.
pTypeLib->Release();
```

Code below shows how to get a pointer to the interface through which GUI can call the calculation functions. First the interface pointer called pUnknown is acquired from COM object. Then calling QueryInterface of pUnknown provides the desired custom interface pointer. This is implemented in Document class of GUI.

```
CLSID           clsid;        // uuid for the COM object
LPUNKNOWN       pUnknown;     // pointer to IUnknown interface
Iobject1*       pTest;        // pointer to the customer interface
HRESULT         hr;           // handle of results

clsid = CLSID_object1;

// structure to store information of one or more than one interface pointer
MULTI_QI mqi[]={ {&IID_Iobject1, NULL , hr} };
COSERVERINFO srvinfo = {0, remote_name, NULL, 0};
// try to get the interface object and store it in mqi.
hr = CoCreateInstanceEx(clsid, NULL,
        /*CLSCTX_REMOTE_SERVER*/CLSCTX_INPROC_SERVER,
        &srvinfo, 1, mqi);

// get interface pointer to IUnknown
if (SUCCEEDED(hr))
{
        // pointer to IUnknown is always stored as the first element of mqi.
        if (SUCCEEDED(mqi[0].hr))
                pUnknown=mqi[0].pItf;
        else
        {
                AfxMessageBox("failed to get IUnknown");
                return;
        }
}
```

```
// cannot get information of interface object
else
{
        AfxMessageBox("failed to connect to CObject");
        return;
}

// get the customer interface pointer
hr = pUnknown->QueryInterface(IID_Iobject1, (void **) &pTest);

// hr is failure return code
if (!SUCCEEDED(hr))
{
        AfxMessageBox("failed to get pTest");
        pUnknown->Release();
        return;
}
pUnknown->Release();
```

Using the interface pointer (pTest) we can call all the functions implemented in the component.

3.2. Input and Output Data Sources

The sole purpose of the COM-based GUI is to collect input and to present the output. This GUI system is capable of handling different formats of input and output data so that user can manipulate data more conveniently.

3.2.1. Input Data Sources

Text files and relational data files are acceptable to the system as input. For text file, MFC provides CFile class to read the data. The process of creating an input stream from input file through Cfile class is illustrated below.

```
// CfileDialog is system defined file open dialog
CFileDialog filedlg(TRUE, "*", "*.dat");
CFile ftemp;
CString filename;

// user click ok button in the dialog
if (filedlg.DoModal() ==IDOK)
{
        // get the complete file name including the path from the dialog
        filename=filedlg.GetPathName();
}
else
        return;

// change the GUI window's title to the name of selected file
SetTitle(filename);

// create the input stream for the input file
fstream fIn((const char*)filename, ios::in);

if (fIn.fail())  {
        cerr<<"can not open the input file!"<<filename<<endl;
        exit(1);
}

unsigned row, col;
// read data from input stream.
fIn>>row>>col;
```

For relational database file, MFC provides CDaoDatabase and CdaoRecordset classes to deal with the data. The following code shows how MFC database classes, CdaoDatabase and CdaoRecordset help establish session with relational database and open the input data table for later use.

```
CDaoDatabase m_database;
CDaoRecordset* m_pRecordset;

// select database files in the dialog
CFileDialog dlg(TRUE, ".mdb", "*.mdb");
if (dlg.DoModal() == IDCANCEL)
        return;
// get complete file name for the database
```

```
m_strDatabase = dlg.GetPathName();

try
{
        // open the database
        m_database.Open(m_strDatabase, FALSE, TRUE);
}
// catch the exception
catch (CDaoException* e)
{
        e->Delete();
        return;
 }

m_strDatabase = m_database.GetName();

if (m_strQuery.IsEmpty())
{
        // select a table in the database through the dialog
        CTableSelect tableDlg(&m_database);
        if (tableDlg.DoModal() != IDOK)
        {
                 m_database.Close();  // escape route
                return;
        }

        // copy the sql query statement to m_strQuery
        m_strQuery.Format("select * from %s", tableDlg.m_strSelection);
}

// create a resultset for the opened database
m_pRecordset = new CDaoRecordset(&m_database);
 try
{
        // put the results of the query statement in the resultset
        m_pRecordset->Open(dbOpenDynaset, m_strQuery, dbReadOnly);
 }
catch (CDaoException* e)
{
        e->Delete();
         return;
 }

// move the pointer to the end of the recordset in order to count the number of
  records.
if (!m_pRecordset->IsBOF())
```

```
{
        m_pRecordset->MoveLast(); // to validate record count
}
m_nRowCount = m_pRecordset->GetAbsolutePosition();
```

For remote text file, MFC provides CInternetSession class to connect to remote URL(Universal Resource Locator) and read data from the input file. Following code shows how CinternetSession class connects remote site file by opening URL and returns a file pointer to handle input data.

```
CInternetSession        session("My Session");
CStdioFile*             pFile = NULL;
Cstring                 url = dlg.m_URL;     // get url from file-open-dialog.

try
{
        // open the URL using the CInternetSession and get a pointer to the remote
          file
        pFile = session.OpenURL(url);
        unsigned row, col;

        // read data in rows
        pFile->ReadString(temp);
        sscanf(temp, "%d %d", &row, &col);
}
// failed to contact remote URL. Catch the exception
catch (CInternetException* pE)
{
        AfxMessageBox("failed to connect to the url");
        return;
}

// close the CInternetSession
session.Close();
```

Since the GUI only collects the input data, it does not provide storage for it. Back-end COM object will provide the storage for the data. The storage class is called Table that is defined in COM object. Following code shows how to transfer the collected input data from GUI client to COM object. On client side input data is temporarily saved in memory then transferred to server object side for constructing Table object. Memory for temporary input data is released then.

```
// pointer to the interface
Iobject1*       pTest;

// initialize the instance data of the COM object
pTest->Initialize();

// creates a new Table instance
pTest->NewTable(row, col, (double)0.0);

double *temp_row;
temp_row = new double[col];

// read the data from input stream and then transfer the data to COM object
for (unsigned j=0; j<row; j++)
{
        for (unsigned i=0; i < col; i++)
        {
                fIn>>*(temp_row+i);
        }
        pTest->WriteRow(temp_row, j);       // transfer the input data in rows.
}

// reclaim the memory
delete [] temp_row;
```

To minimize the network overhead especially when transferring large amount of data, this GUI system moves the data in rows as shown above instead of individual elements.

On COM object side, the input data transferred from GUI is stored in a new instance of Table. WriteRow method accepts an array of data and stores it in the instance of Table. To minimize the network traffic, a row of data is transferred each time as shown below.

```
STDMETHODIMP Cobject1::WriteRow(double *temp_row, int i)
{
        // system generated
        AFX_MANAGE_STATE(AfxGetStaticModuleState())

        int rowNum, colNum;
        rowNum = data->GetRow();
        colNum = data->GetCol();

        (*data)[i].input(temp_row);

        return S_OK;
}
```

3.2.2. Output Data formats

There are two types of output files compatible with the system, text file and csv file. Both of them are available through menu items, Save and Save As. The GUI system retrieves the contents of CrichEditView and saves them in appropriate files. The difference between csv and text file is that csv is comma delimited instead of space delimited. The text file is space delimited. The advantage of csv file is that it can be recognized by some Windows applications such as Excel.

We show how the CrichEditView opens a file and saves the contents of the view to the file using the following code segment:

```
        CFile                   file;

        // get CrichEditCtrl through CRichEditView
        CRichEditCtrl&          edit = GetRichEditCtrl();

        // open the file to write
        if (!file.Open(filename, CFile::modeCreate | CFile::modeWrite |
                CFile::modeNoTruncate, &e))
        {
                // for debug purpose
                #ifdef _DEBUG
                        afxDump << "Unable to open file" << e.m_cause << "\n";
                #endif

                // if the file cannot be opened, return with error information
```

```
            filename = "Can not open the file: " + filename;
            AfxMessageBox(filename);
            return;
    }

    // count the number of lines shown in the view
    int line_count = edit.GetLineCount();
    for (int i = 0; i < line_count; i++)
    {
            edit.GetLine(i, buf, 256);
            buffer.Format(buf);

            ...
            // copy the contents in lines
            file.Write(buffer, buffer.GetLength());
            strcpy(buf, "");
    }
```

3.3. Data Exchange

Data exchange is very important in modern window applications. Different applications can directly share the data. More important is that one application can utilize the functions of another application with minimal overhead.

This thesis makes it possible to have the GUI exchanging data with Microsoft Excel since the input/output data and intermediate results are stored in a matrix that actually is a table. Excel provides many convenient functions to manipulate data in its datasheet including saving file in html format.

A menu item / toolbar of "Export to Excel" has been added to the GUI to fulfil the functionality of dada exchange with Excel. The design strategy is to create an instance of

Excel application through the wrapper classes of Microsoft Excel, excel.cpp, excel.h. The code below shows how the COM-based GUI starts the Excel application and makes the window visible.

```
    _Application    excelApp;       // one of the wrapper classes for Excel
    boolean         visible;

    if (!excelApp)
    {
            // create a dispatch for excelApp
            if (!excelApp.CreateDispatch("Excel.Application"))
            {
                    AfxMessageBox("Couldn't start Excel.");
                    return;
            }
    }
    else
    {
            // check the visible status of excelApp
            visible = excelApp.GetVisible();
            if (!visible)
            {
                    excelApp = NULL;
                    delete excelApp;
                    if (!excelApp.CreateDispatch("Excel.Application"))
                    {
                            AfxMessageBox("Couldn't start Excel.");
                            return;
                    }
            }
    }

    // makes the window visible
    excelApp.SetVisible(TRUE);
    // set the window size as maximized
    excelApp.SetWindowState(3);         // 1-normal 2-minimized 3-
                                           maximized
    excelApp.SetUserControl(TRUE);

}
```

CHAPTER IV

Results and Discussion

We use several sets of testing data including the one published in[6]. The data has been stored in text file (local and remote) and relational database file to test the system.

Figures 5 through 17 show the user interface of the system. Figure 5 illustrates the screen when user first starts the application. Figure 6 and figure 7 show how to load an input file through the menu. Figure 8 and figure 9 illustrate how to open a relational database file as input source. Figure 10 illustrates how to connect to an URL and open the remote input file. Figure 11 illustrates how to open an Excel workbook file (extension xls). Figure 12 shows how to save the contents of the view as original input file (same file name). Figure 13 shows how to save as file (system generated file name). Figure 14 shows the worksheet containing the exported results from the system. Figures 15, 16, 17, and 18 illustrate the functionality of copy, cut and paste. Figure 19 shows the print dialog to print certain pages at selected printer. Figure 20 illustrates what print-preview looks like for the contents of current view. Figure 21 shows print setup for paper size and orientation.

This system separates GUI from calculation logic that is contained in COM component. Each feature that we add to the system needs to be analyzed and divided into two pieces, each of which goes to either GUI or COM component. For GUI part, dialog design and class wizard provide tools to complete the implementation. For COM component part, ATL helps accomplishing adding calculation functions. If IDL file, which defines interface functions, changes, both GUI and COM components are to be recompiled.

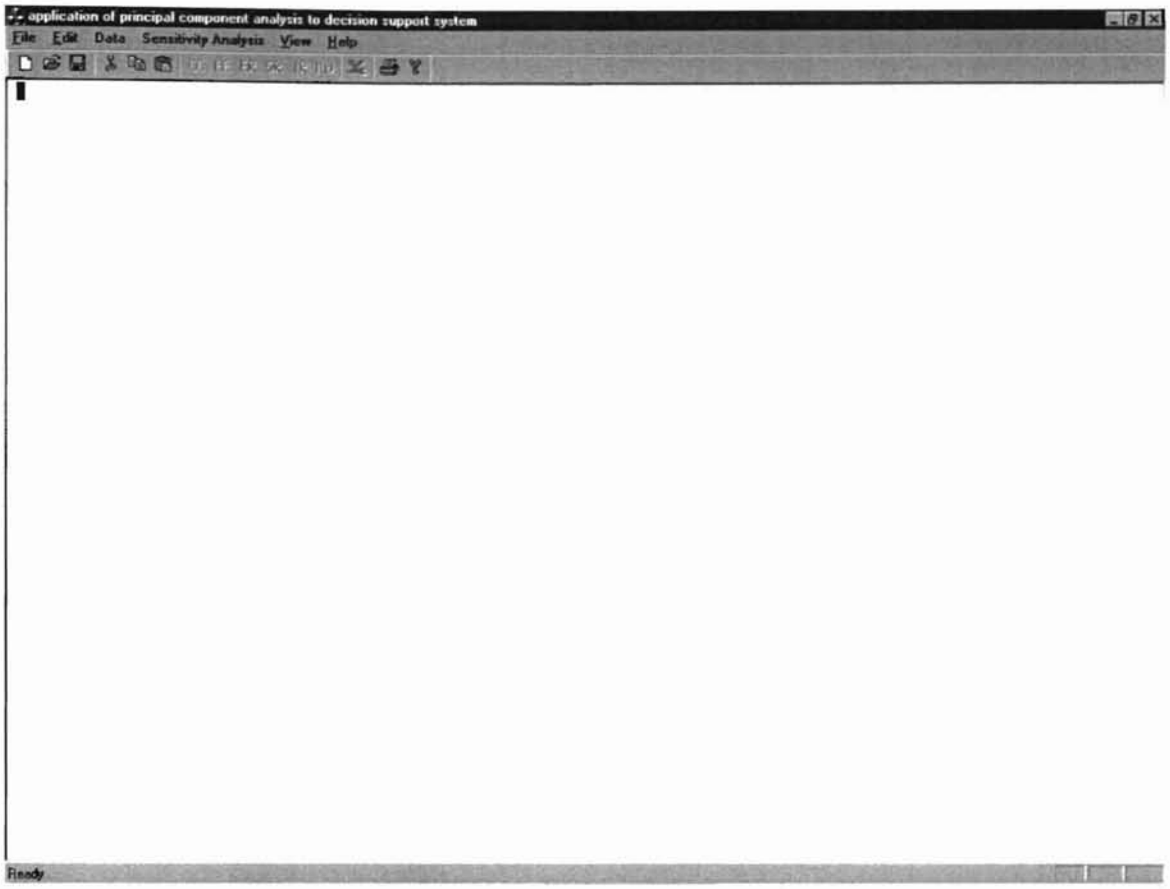

application of principal component analysis to decision support system
File Edit Data Sensitivity Analysis View Help
Ready

Figure 5. Startup screen

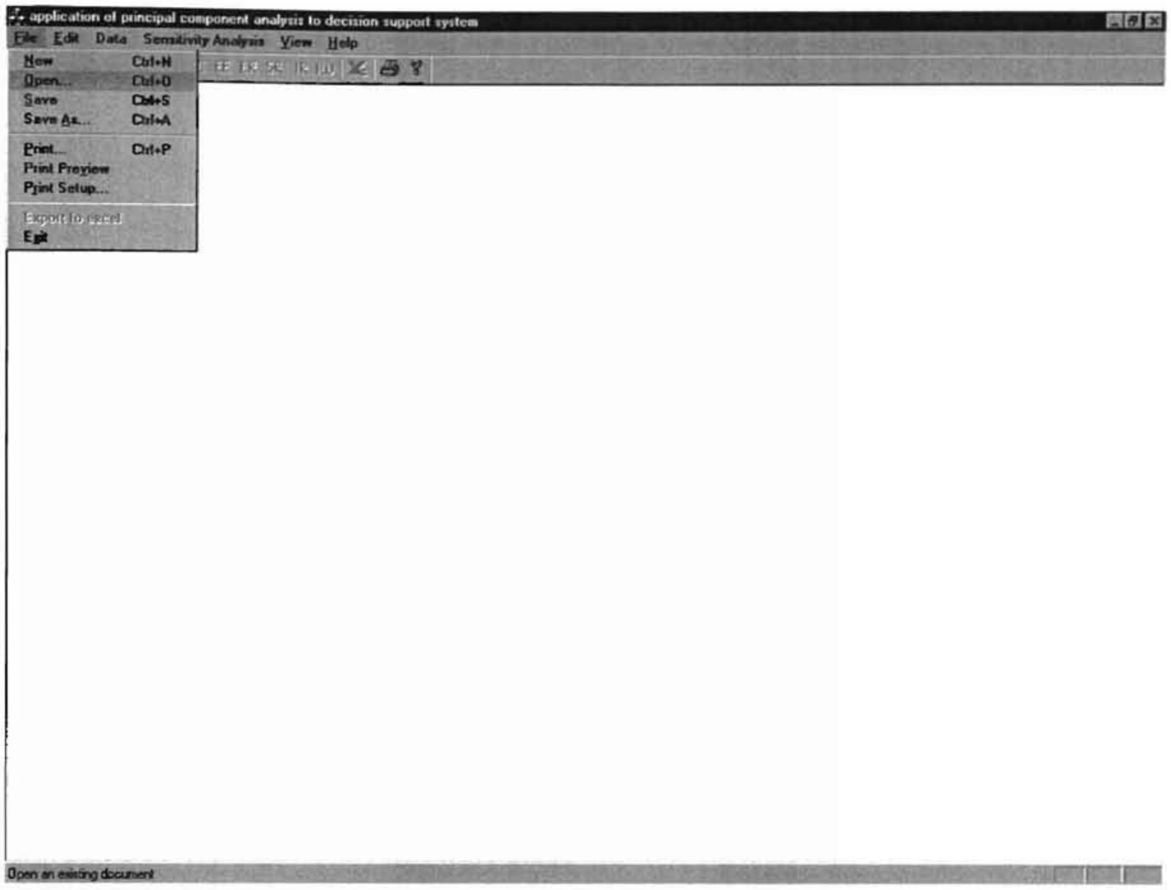

application of principal component analysis to decision support system
File Edit Data Sensitivity Analysis View Help
New Ctrl+N
Open... Ctrl+O
Save Ctrl+S
Save As... Ctrl+A
Print... Ctrl+P
Print Preview
Print Setup...
Export to excel
Exit
Open an existing document

Figure 6. Open input file through menu

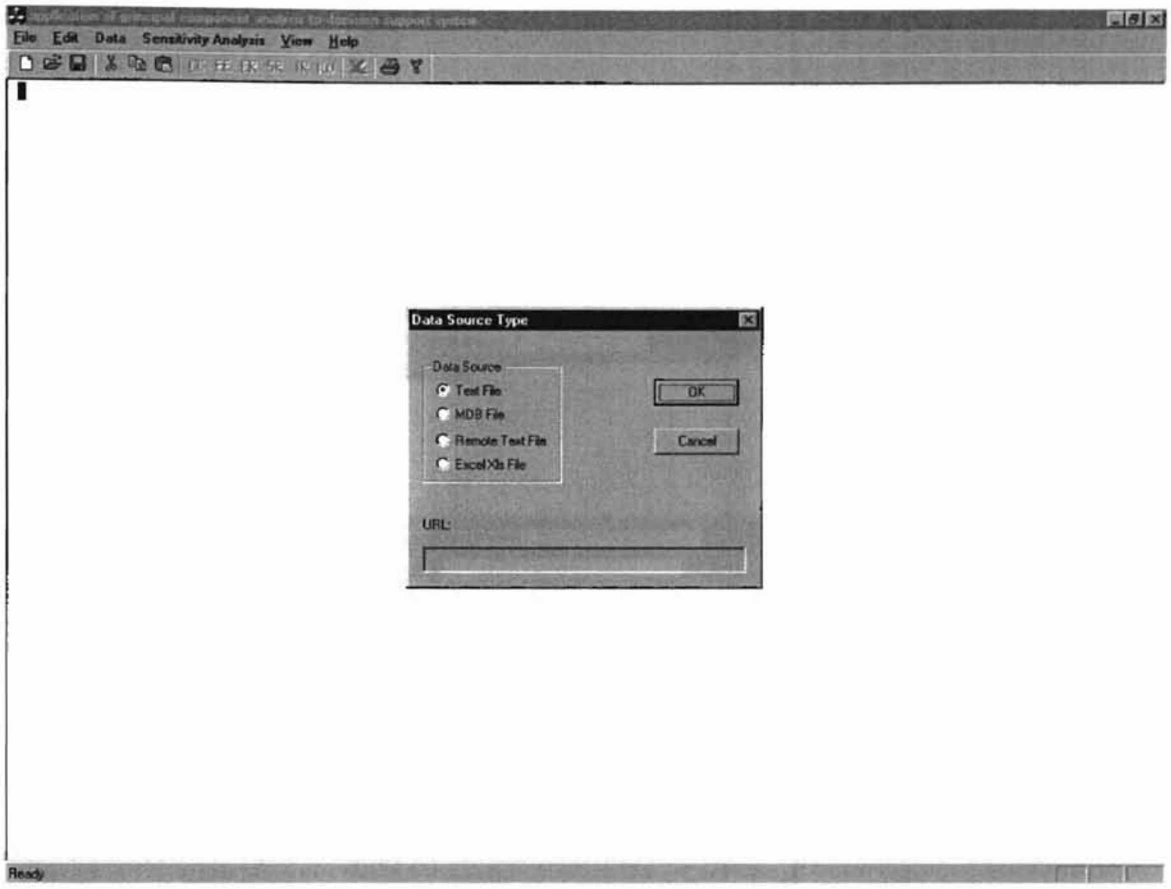
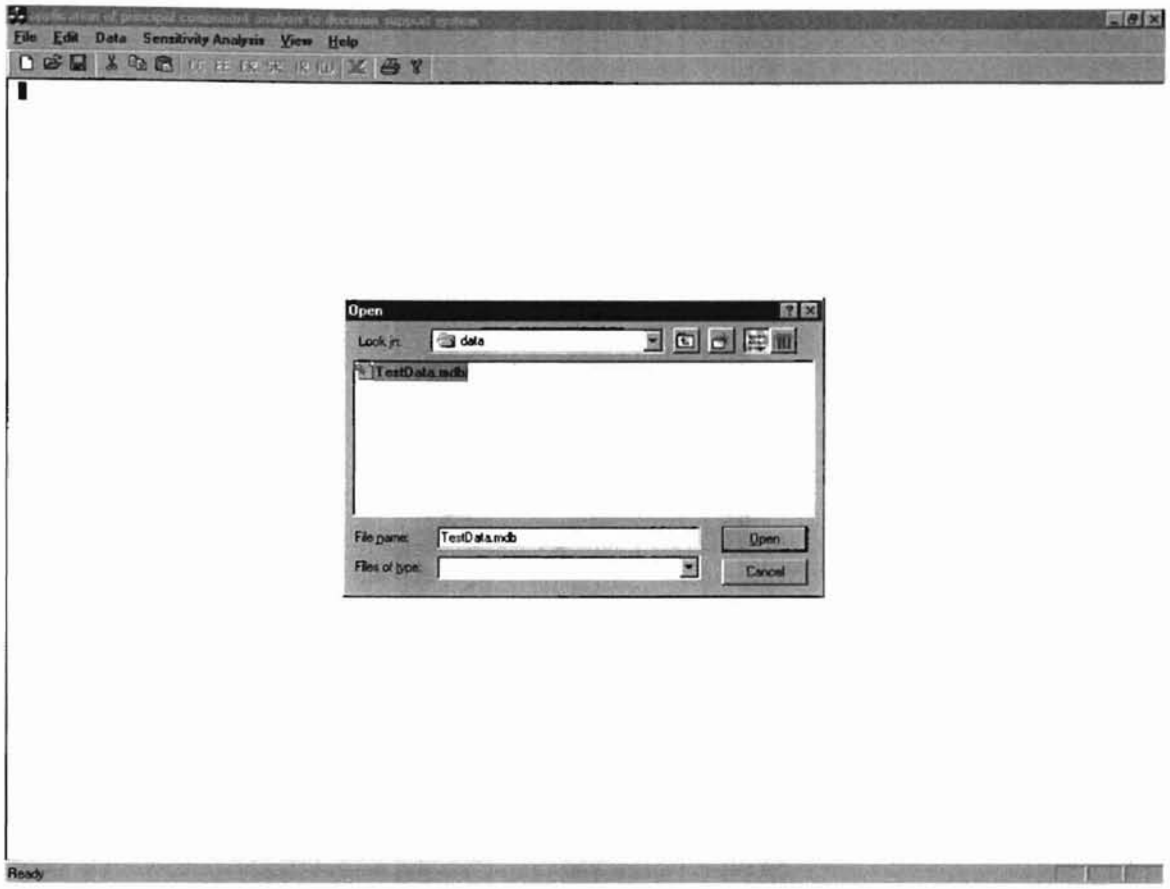


Figure 7. Open file dialog



File name: TestData.mdb

Files of type:

Figure 8. Open database file dialog

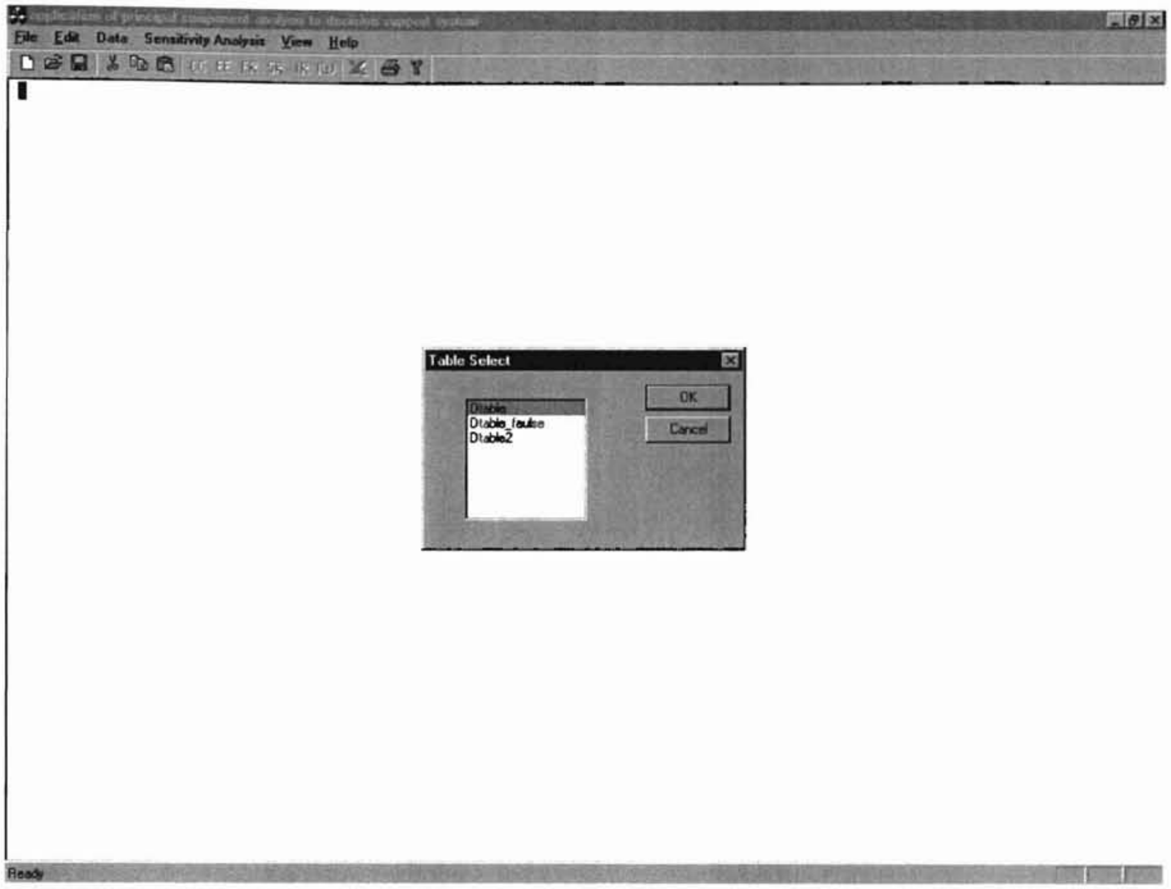
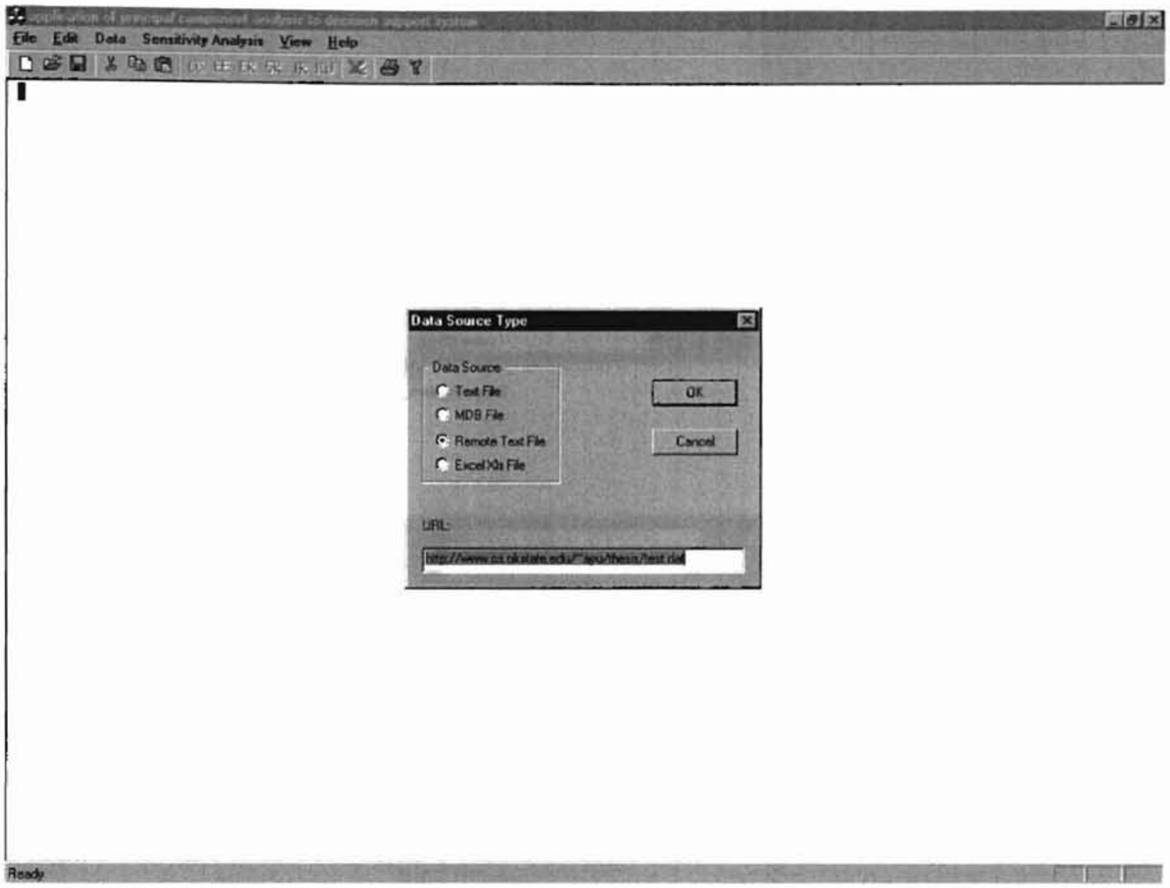


Figure 9. Select database table dialog


application of principal component analysis to decision support system
File Edit Data Sensitivity Analysis View Help
Data Source Type
OK
Cancel
Ready

Data Source: ( ) Text File ( ) MDB File [✓ Remote Text File] ( ) Excel Xls File

URL: http://www.cs.okstate.edu/~apu/thesis/test.dat

Figure 10. Open remote site input file

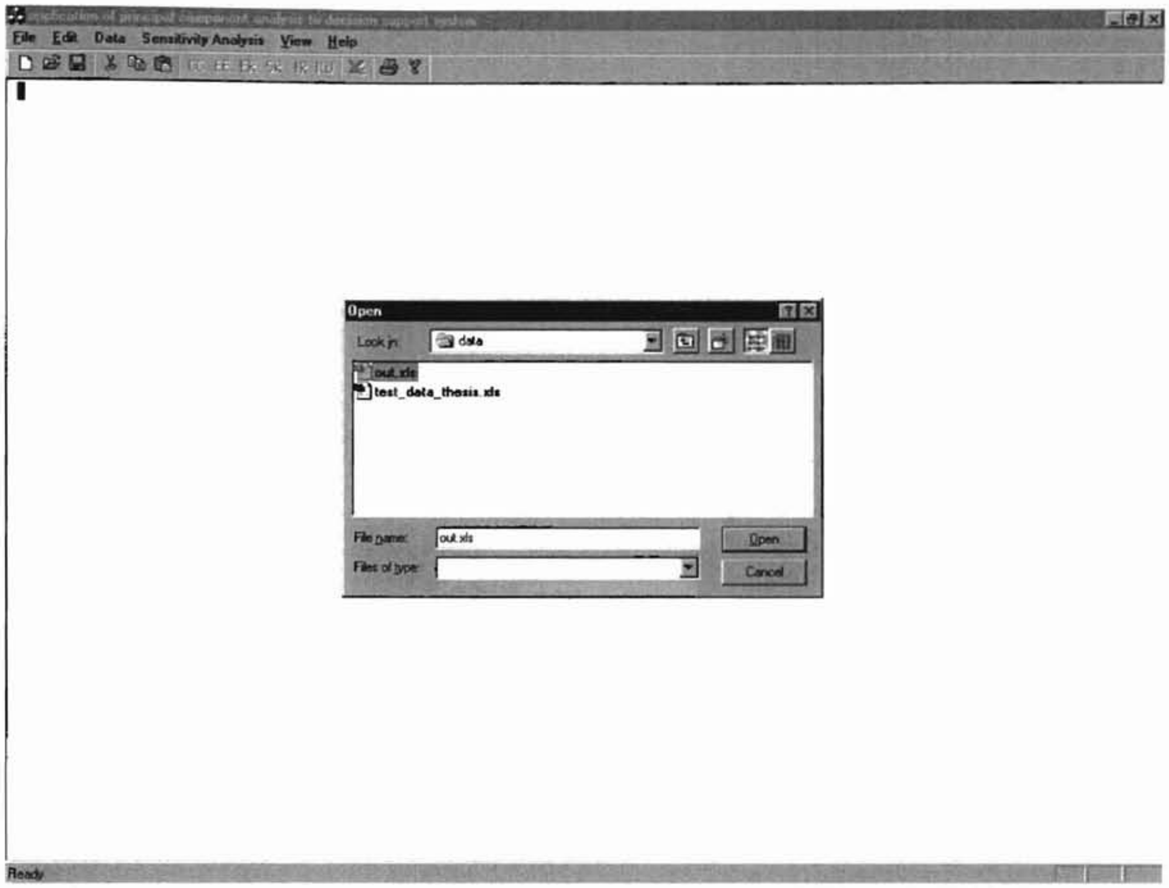
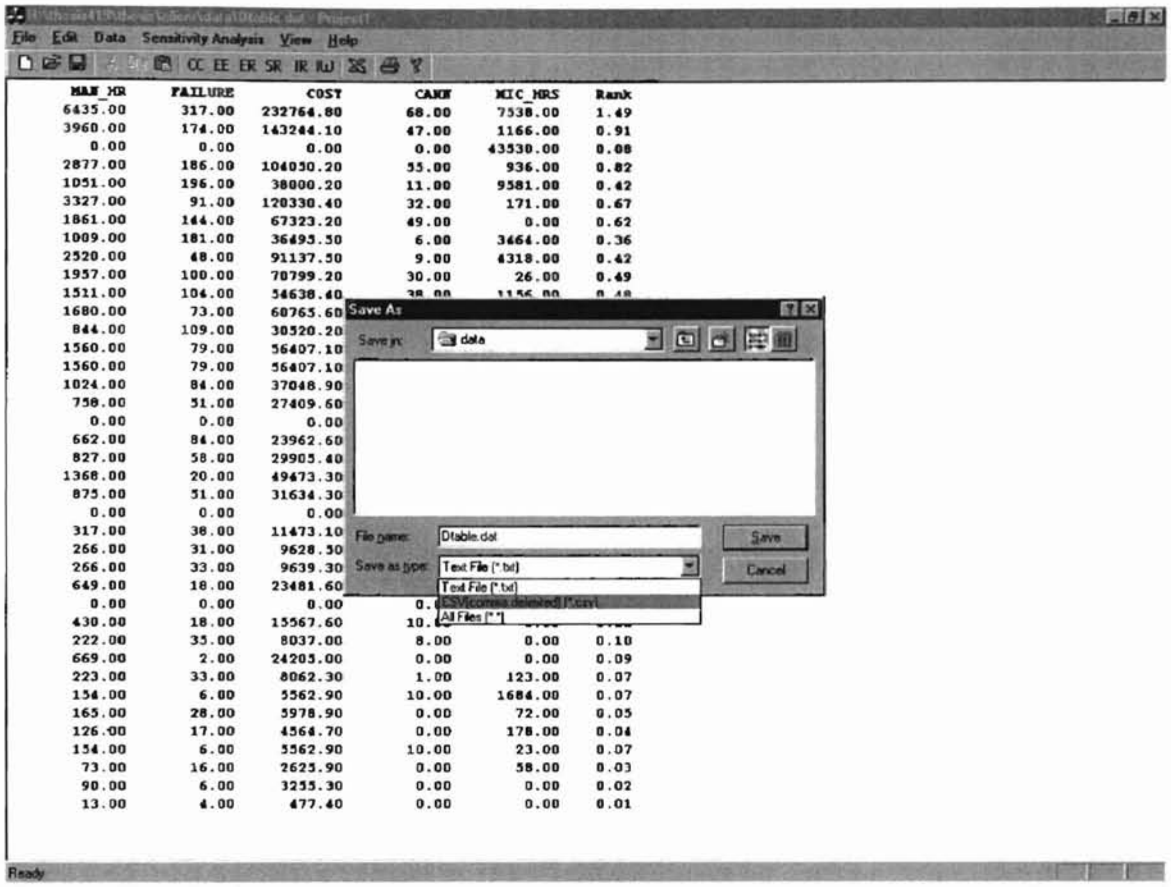


Figure 11. Open Excel Workbook file



```
 MAN_HR   FAILURE       COST      CANN    MIC_HRS   Rank
6435.00    317.00  232764.80     68.00    7538.00   1.49
3960.00    174.00  143244.10     47.00    1166.00   0.91
   0.00      0.00       0.00      0.00   43530.00   0.08
2877.00    186.00  104030.20     55.00     936.00   0.82
1051.00    196.00   38000.20     11.00    9581.00   0.42
3327.00     91.00  120330.40     32.00     171.00   0.67
1861.00    144.00   67323.20     49.00       0.00   0.62
1009.00    181.00   36495.50      6.00    3464.00   0.36
2520.00     48.00   91137.50      9.00    4318.00   0.42
1957.00    100.00   70799.20     30.00      26.00   0.49
1511.00    104.00   54638.40     38.00    1156.00   0.48
1680.00     73.00   60765.60
 844.00    109.00   30520.20
1560.00     79.00   56407.10
1560.00     79.00   56407.10
1024.00     84.00   37048.90
 758.00     51.00   27409.60
   0.00      0.00       0.00
 662.00     84.00   23962.60
 827.00     58.00   29905.40
1368.00     20.00   49473.30
 875.00     51.00   31634.30
   0.00      0.00       0.00
 317.00     38.00   11473.10
 266.00     31.00    9628.50
 266.00     33.00    9639.30
 649.00     18.00   23481.60
   0.00      0.00       0.00      0.
 430.00     18.00   15567.60     10.
 222.00     35.00    8037.00      8.00       0.00   0.10
 669.00      2.00   24205.00      0.00       0.00   0.09
 223.00     33.00    8062.30      1.00     123.00   0.07
 154.00      6.00    5562.90     10.00    1684.00   0.07
 165.00     28.00    5978.90      0.00      72.00   0.05
 126.00     17.00    4564.70      0.00     178.00   0.04
 154.00      6.00    5562.90     10.00      23.00   0.07
  73.00     16.00    2625.90      0.00      58.00   0.03
  90.00      6.00    3255.30      0.00       0.00   0.02
  13.00      4.00     477.40      0.00       0.00   0.01
```

Save As

Save in: data

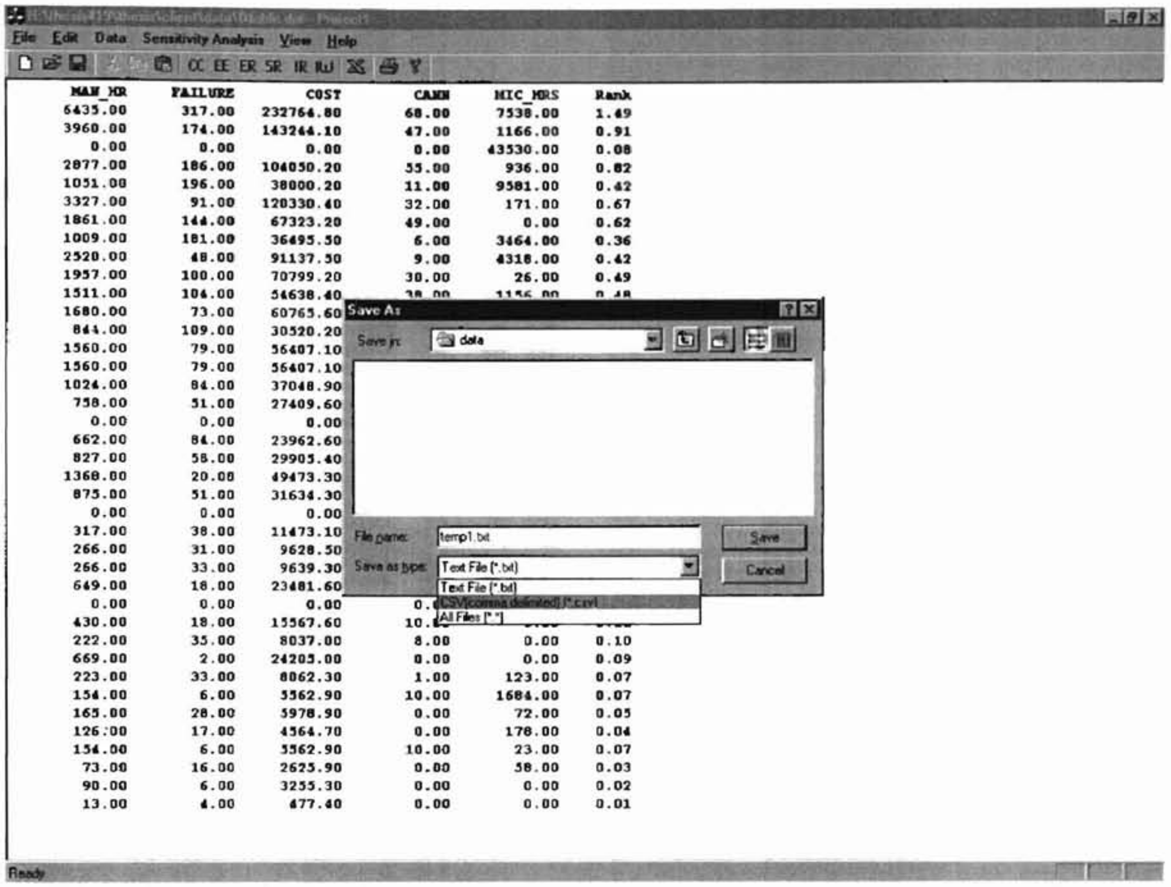
File name: Dtable.dat

Save as type: Text File (*.txt)

Text File (*.txt)
CSV(comma delimited) (*.csv)
All Files (*.*)

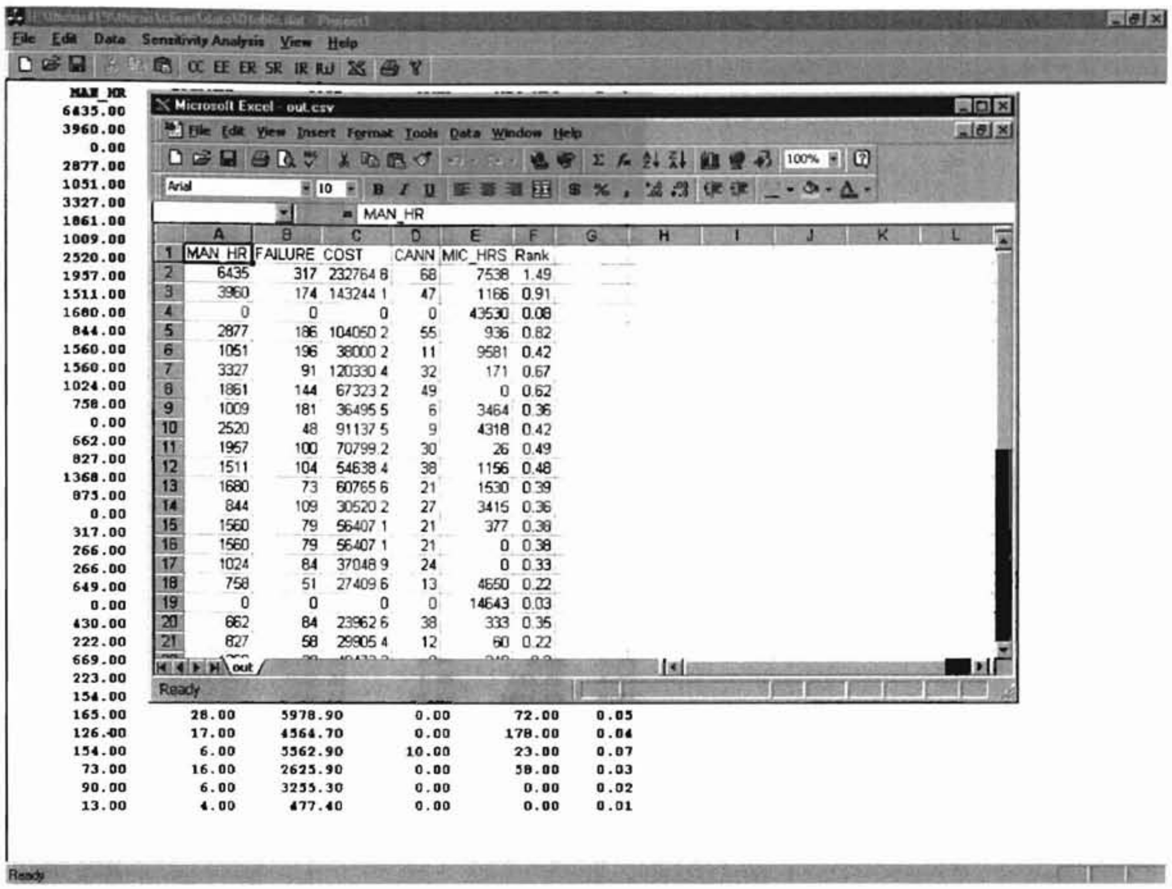
Save
Cancel

Figure 12. Save input file



MAN_HR	FAILURE	COST	CANN	MIC_HRS	Rank
6435.00	317.00	232764.80	68.00	7538.00	1.49
3960.00	174.00	143244.10	47.00	1166.00	0.91
0.00	0.00	0.00	0.00	43530.00	0.08
2877.00	186.00	104050.20	55.00	936.00	0.82
1051.00	196.00	38000.20	11.00	9581.00	0.42
3327.00	91.00	120330.40	32.00	171.00	0.67
1861.00	144.00	67323.20	49.00	0.00	0.62
1009.00	181.00	36495.50	6.00	3464.00	0.36
2520.00	48.00	91137.50	9.00	4318.00	0.42
1957.00	100.00	70799.20	30.00	26.00	0.49
1511.00	104.00	54638.40	38.00	1156.00	0.48
1680.00	73.00	60765.60			
844.00	109.00	30520.20			
1560.00	79.00	56407.10			
1560.00	79.00	56407.10			
1024.00	84.00	37048.90			
758.00	51.00	27409.60			
0.00	0.00	0.00			
662.00	84.00	23962.60			
827.00	58.00	29905.40			
1368.00	20.00	49473.30			
875.00	51.00	31634.30			
0.00	0.00	0.00			
317.00	38.00	11473.10			
266.00	31.00	9628.50			
266.00	33.00	9639.30			
649.00	18.00	23481.60			
0.00	0.00	0.00	0.		
430.00	18.00	15567.60	10.		
222.00	35.00	8037.00	8.00	0.00	0.10
669.00	2.00	24205.00	0.00	0.00	0.09
223.00	33.00	8062.30	1.00	123.00	0.07
154.00	6.00	5562.90	10.00	1684.00	0.07
165.00	28.00	5978.90	0.00	72.00	0.05
126.00	17.00	4564.70	0.00	178.00	0.04
154.00	6.00	5562.90	10.00	23.00	0.07
73.00	16.00	2625.90	0.00	58.00	0.03
90.00	6.00	3255.30	0.00	0.00	0.02
13.00	4.00	477.40	0.00	0.00	0.01

Figure 13. Save as temporary file



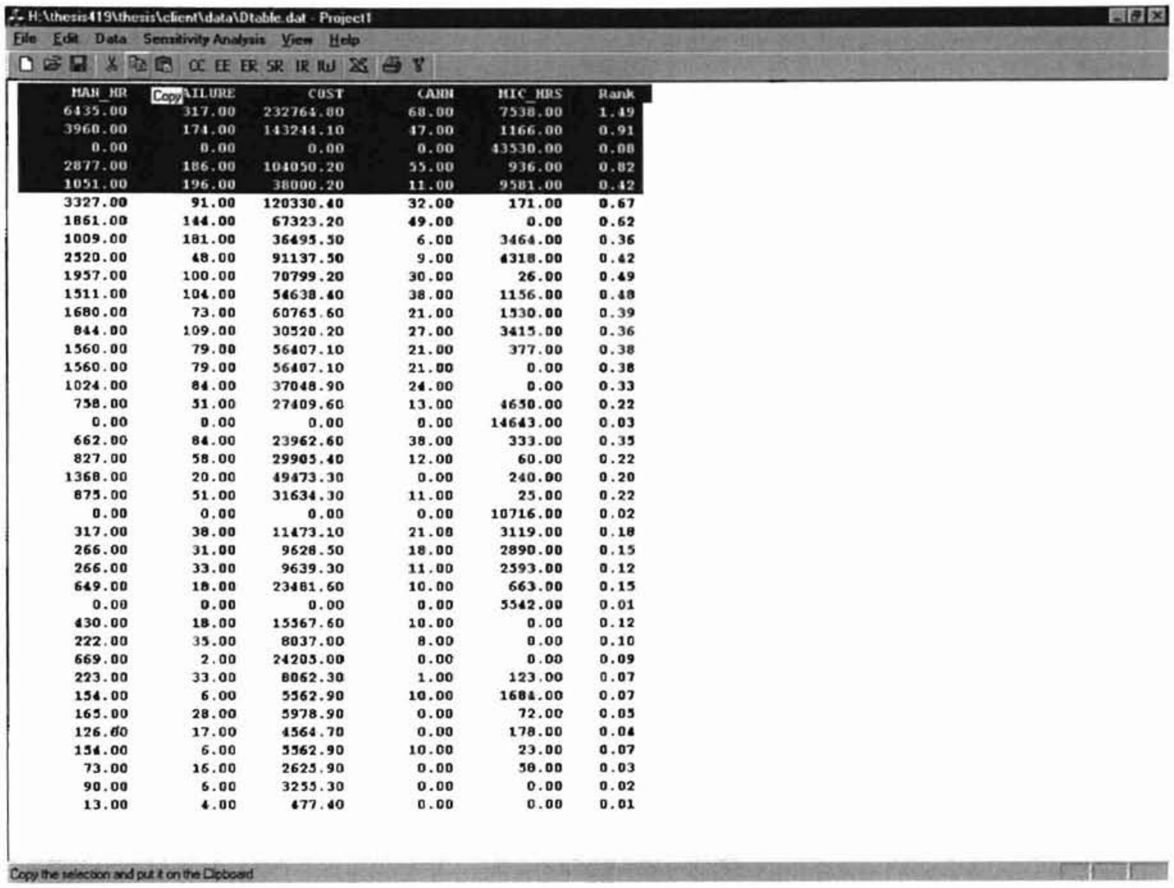
File Edit Data Sensitivity Analysis View Help
Microsoft Excel - out.csv
File Edit View Insert Format Tools Data Window Help
Arial
10
100%
out
Ready

```
 MAN_HR
6435.00
3960.00
   0.00
2877.00
1051.00
3327.00
1861.00
1009.00
2520.00
1957.00
1511.00
1680.00
 844.00
1560.00
1560.00
1024.00
 758.00
   0.00
 662.00
 827.00
1368.00
 875.00
   0.00
 317.00
 266.00
 266.00
 649.00
   0.00
 430.00
 222.00
 669.00
 223.00
 154.00
 165.00     28.00    5978.90      0.00     72.00    0.05
 126.00     17.00    4564.70      0.00    178.00    0.04
 154.00      6.00    5562.90     10.00     23.00    0.07
  73.00     16.00    2625.90      0.00     58.00    0.03
  90.00      6.00    3255.30      0.00      0.00    0.02
  13.00      4.00     477.40      0.00      0.00    0.01
```

= MAN_HR

	A	B	C	D	E	F
1	MAN_HR	FAILURE	COST	CANN	MIC_HRS	Rank
2	6435	317	232764.8	68	7538	1.49
3	3960	174	143244.1	47	1166	0.91
4	0	0	0	0	43530	0.08
5	2877	186	104060.2	55	936	0.82
6	1051	196	38000.2	11	9581	0.42
7	3327	91	120330.4	32	171	0.67
8	1861	144	67323.2	49	0	0.62
9	1009	181	36495.5	6	3464	0.36
10	2520	48	91137.5	9	4318	0.42
11	1957	100	70799.2	30	26	0.49
12	1511	104	54638.4	38	1156	0.48
13	1680	73	60765.6	21	1530	0.39
14	844	109	30520.2	27	3415	0.36
15	1560	79	56407.1	21	377	0.38
16	1560	79	56407.1	21	0	0.38
17	1024	84	37048.9	24	0	0.33
18	758	51	27409.6	13	4650	0.22
19	0	0	0	0	14643	0.03
20	662	84	23962.6	38	333	0.35
21	827	58	29905.4	12	60	0.22

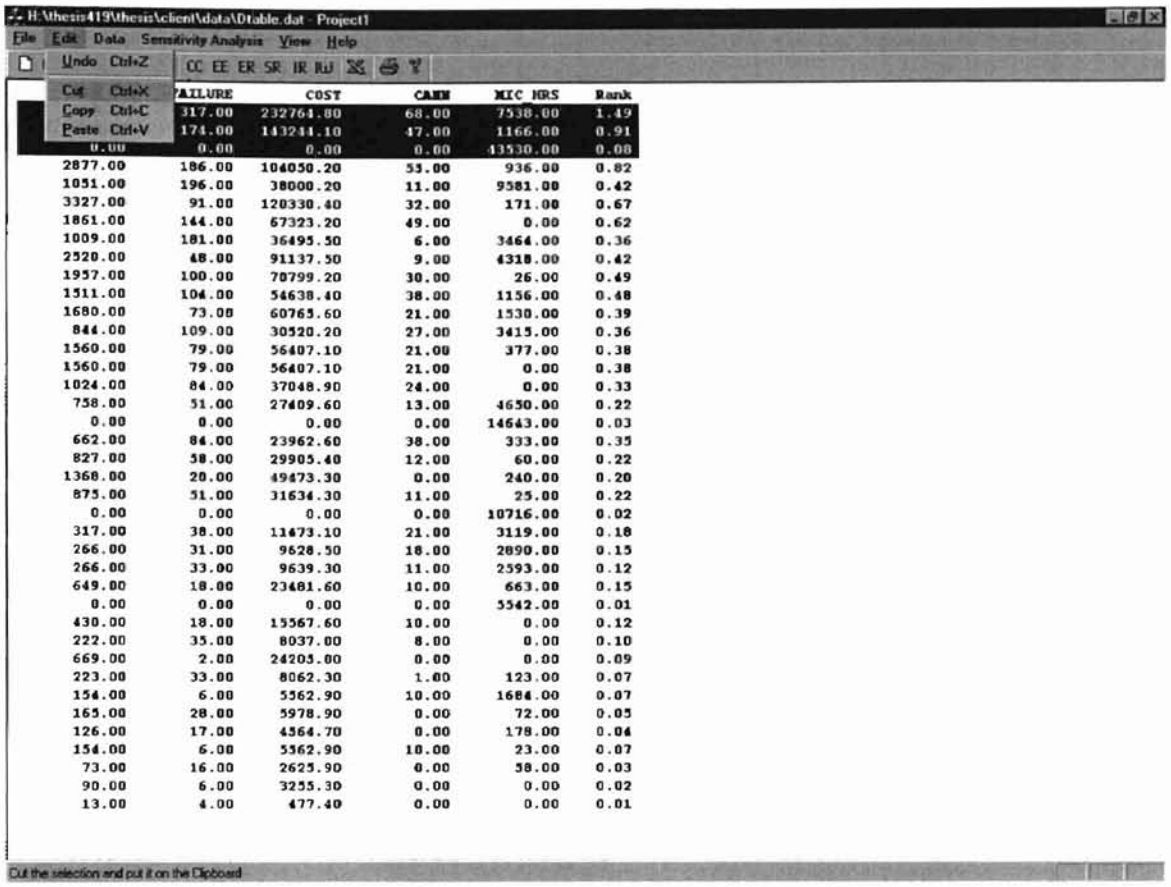
Figure 14. Export to Excel


H:\thesis419\thesis\client\data\Dtable.dat - Project1
File Edit Data Sensitivity Analysis View Help
CC EE ER SR IR IIJ
Copy

MAN_HR	AILURE	COST	CANN	MIC_HRS	Rank
6435.00	317.00	232764.80	68.00	7538.00	1.49
3960.00	174.00	143244.10	47.00	1166.00	0.91
0.00	0.00	0.00	0.00	43530.00	0.08
2877.00	186.00	104050.20	55.00	936.00	0.82
1051.00	196.00	38000.20	11.00	9581.00	0.42
3327.00	91.00	120330.40	32.00	171.00	0.67
1861.00	144.00	67323.20	49.00	0.00	0.62
1009.00	181.00	36495.50	6.00	3464.00	0.36
2520.00	48.00	91137.50	9.00	4318.00	0.42
1957.00	100.00	70799.20	30.00	26.00	0.49
1511.00	104.00	54638.40	38.00	1156.00	0.48
1680.00	73.00	60765.60	21.00	1530.00	0.39
844.00	109.00	30520.20	27.00	3415.00	0.36
1560.00	79.00	56407.10	21.00	377.00	0.38
1560.00	79.00	56407.10	21.00	0.00	0.38
1024.00	84.00	37048.90	24.00	0.00	0.33
758.00	51.00	27409.60	13.00	4650.00	0.22
0.00	0.00	0.00	0.00	14643.00	0.03
662.00	84.00	23962.60	38.00	333.00	0.35
827.00	58.00	29905.40	12.00	60.00	0.22
1368.00	20.00	49473.30	0.00	240.00	0.20
875.00	51.00	31634.30	11.00	25.00	0.22
0.00	0.00	0.00	0.00	10716.00	0.02
317.00	38.00	11473.10	21.00	3119.00	0.18
266.00	31.00	9628.50	18.00	2890.00	0.15
266.00	33.00	9639.30	11.00	2593.00	0.12
649.00	18.00	23481.60	10.00	663.00	0.15
0.00	0.00	0.00	0.00	5542.00	0.01
430.00	18.00	15567.60	10.00	0.00	0.12
222.00	35.00	8037.00	8.00	0.00	0.10
669.00	2.00	24205.00	0.00	0.00	0.09
223.00	33.00	8062.30	1.00	123.00	0.07
154.00	6.00	5562.90	10.00	1684.00	0.07
165.00	28.00	5978.90	0.00	72.00	0.05
126.00	17.00	4564.70	0.00	178.00	0.04
154.00	6.00	5562.90	10.00	23.00	0.07
73.00	16.00	2625.90	0.00	58.00	0.03
90.00	6.00	3255.30	0.00	0.00	0.02
13.00	4.00	477.40	0.00	0.00	0.01

Copy the selection and put it on the Clipboard

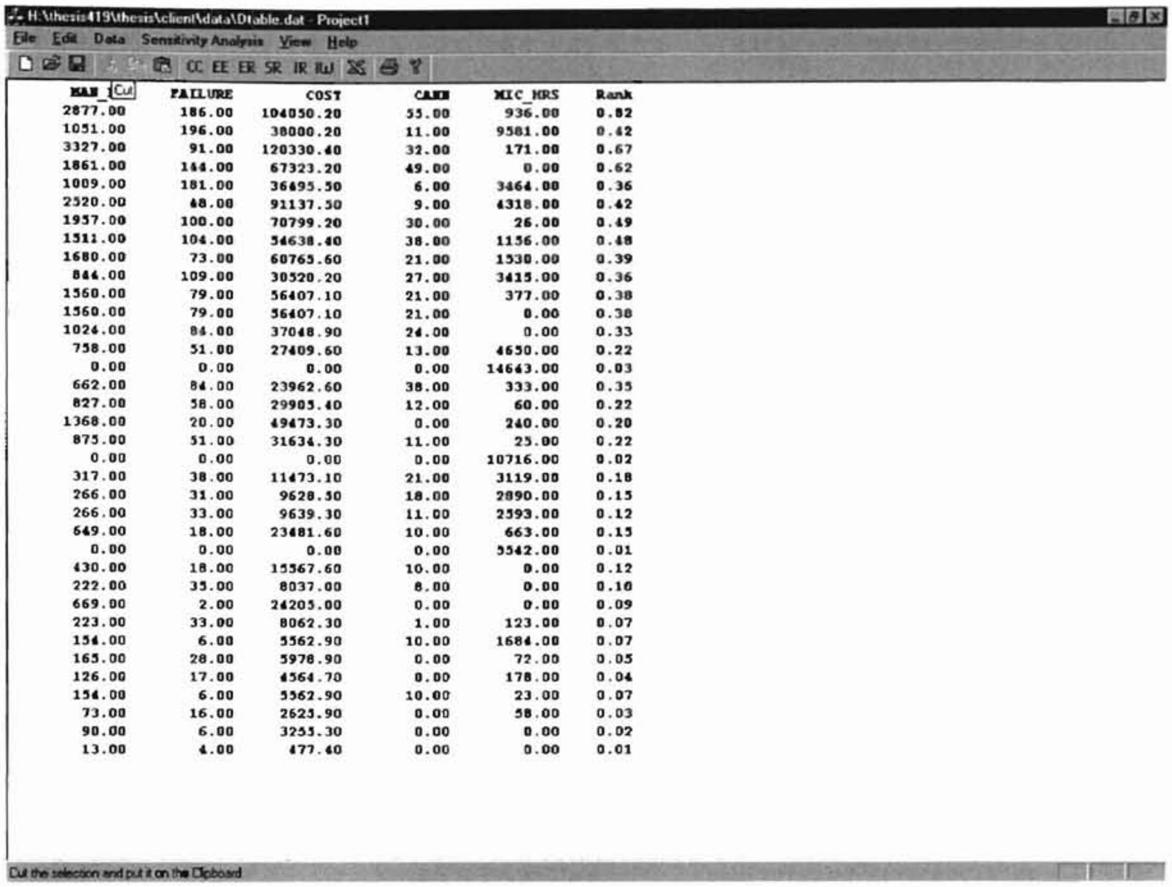
Figure 15. Copy contents of current view



	AILURE	COST	CAIN	MIC_HRS	Rank
	317.00	232764.80	68.00	7538.00	1.49
	174.00	143244.10	47.00	1166.00	0.91
0.00	0.00	0.00	0.00	43530.00	0.08
2877.00	186.00	104050.20	53.00	936.00	0.82
1051.00	196.00	38000.20	11.00	9581.00	0.42
3327.00	91.00	120330.40	32.00	171.00	0.67
1861.00	144.00	67323.20	49.00	0.00	0.62
1009.00	181.00	36495.50	6.00	3464.00	0.36
2520.00	48.00	91137.50	9.00	4318.00	0.42
1957.00	100.00	70799.20	30.00	26.00	0.49
1511.00	104.00	54638.40	38.00	1156.00	0.48
1680.00	73.00	60765.60	21.00	1530.00	0.39
844.00	109.00	30520.20	27.00	3415.00	0.36
1560.00	79.00	56407.10	21.00	377.00	0.38
1560.00	79.00	56407.10	21.00	0.00	0.38
1024.00	84.00	37048.90	24.00	0.00	0.33
758.00	51.00	27409.60	13.00	4650.00	0.22
0.00	0.00	0.00	0.00	14643.00	0.03
662.00	84.00	23962.60	38.00	333.00	0.35
827.00	58.00	29905.40	12.00	60.00	0.22
1368.00	20.00	49473.30	0.00	240.00	0.20
875.00	51.00	31634.30	11.00	25.00	0.22
0.00	0.00	0.00	0.00	10716.00	0.02
317.00	38.00	11473.10	21.00	3119.00	0.18
266.00	31.00	9628.50	18.00	2890.00	0.15
266.00	33.00	9639.30	11.00	2593.00	0.12
649.00	18.00	23481.60	10.00	663.00	0.15
0.00	0.00	0.00	0.00	5542.00	0.01
430.00	18.00	15567.60	10.00	0.00	0.12
222.00	35.00	8037.00	8.00	0.00	0.10
669.00	2.00	24205.00	0.00	0.00	0.09
223.00	33.00	8062.30	1.00	123.00	0.07
154.00	6.00	5562.90	10.00	1684.00	0.07
165.00	28.00	5978.90	0.00	72.00	0.05
126.00	17.00	4564.70	0.00	178.00	0.04
154.00	6.00	5562.90	10.00	23.00	0.07
73.00	16.00	2625.90	0.00	58.00	0.03
90.00	6.00	3255.30	0.00	0.00	0.02
13.00	4.00	477.40	0.00	0.00	0.01

Cut the selection and put it on the Clipboard

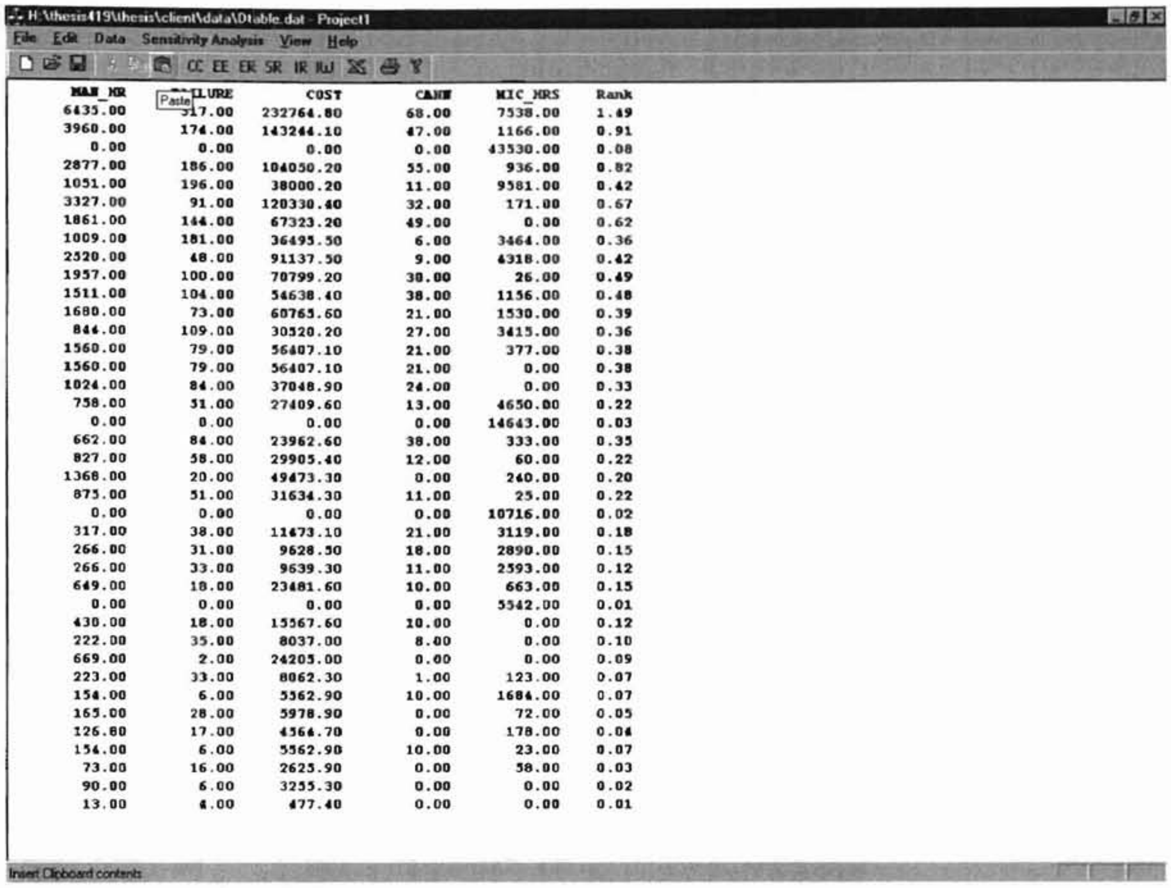
Figure 16. Cut contents of current view



MAN_	FAILURE	COST	CANN	MIC_HRS	Rank
2877.00	186.00	104050.20	55.00	936.00	0.82
1051.00	196.00	38000.20	11.00	9581.00	0.42
3327.00	91.00	120330.40	32.00	171.00	0.67
1861.00	144.00	67323.20	49.00	0.00	0.62
1009.00	181.00	36495.50	6.00	3464.00	0.36
2520.00	48.00	91137.50	9.00	4318.00	0.42
1957.00	100.00	70799.20	30.00	26.00	0.49
1511.00	104.00	54638.40	38.00	1156.00	0.48
1680.00	73.00	60765.60	21.00	1530.00	0.39
844.00	109.00	30520.20	27.00	3415.00	0.36
1560.00	79.00	56407.10	21.00	377.00	0.38
1560.00	79.00	56407.10	21.00	0.00	0.38
1024.00	84.00	37048.90	24.00	0.00	0.33
758.00	51.00	27409.60	13.00	4650.00	0.22
0.00	0.00	0.00	0.00	14643.00	0.03
662.00	84.00	23962.60	38.00	333.00	0.35
827.00	58.00	29905.40	12.00	60.00	0.22
1368.00	20.00	49473.30	0.00	240.00	0.20
875.00	51.00	31634.30	11.00	25.00	0.22
0.00	0.00	0.00	0.00	10716.00	0.02
317.00	38.00	11473.10	21.00	3119.00	0.18
266.00	31.00	9628.50	18.00	2890.00	0.15
266.00	33.00	9639.30	11.00	2593.00	0.12
649.00	18.00	23481.60	10.00	663.00	0.15
0.00	0.00	0.00	0.00	5542.00	0.01
430.00	18.00	15567.60	10.00	0.00	0.12
222.00	35.00	8037.00	8.00	0.00	0.10
669.00	2.00	24205.00	0.00	0.00	0.09
223.00	33.00	8062.30	1.00	123.00	0.07
154.00	6.00	5562.90	10.00	1684.00	0.07
165.00	28.00	5978.90	0.00	72.00	0.05
126.00	17.00	4564.70	0.00	178.00	0.04
154.00	6.00	5562.90	10.00	23.00	0.07
73.00	16.00	2625.90	0.00	58.00	0.03
90.00	6.00	3255.30	0.00	0.00	0.02
13.00	4.00	477.40	0.00	0.00	0.01

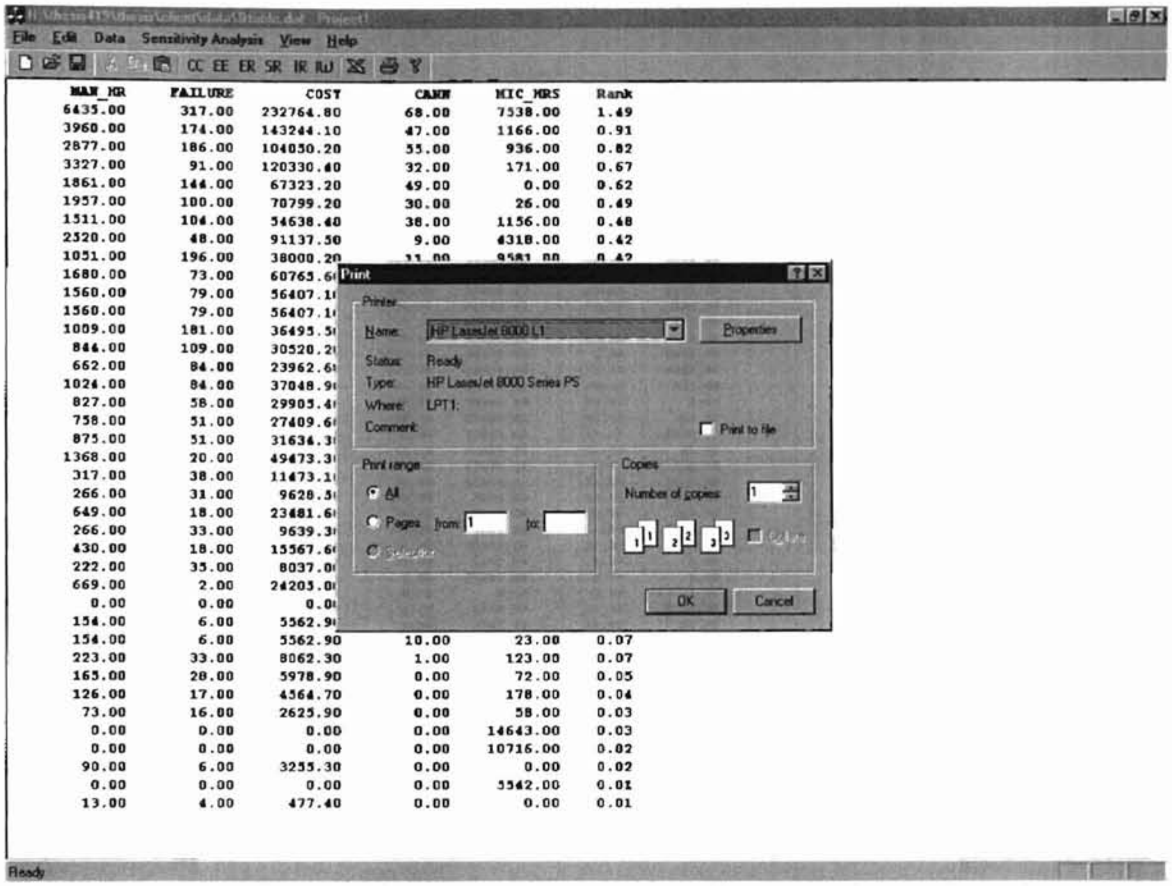
Cut the selection and put it on the Clipboard

Figure 17. After cut contents


H:\thesis419\thesis\client\data\Dtable.dat - Project1
File Edit Data Sensitivity Analysis View Help
CC EE EK SR IR IUJ
Paste
Insert Clipboard contents

MAN_HR	FAILURE	COST	CANN	MIC_HRS	Rank
6435.00	317.00	232764.80	68.00	7538.00	1.49
3960.00	174.00	143244.10	47.00	1166.00	0.91
0.00	0.00	0.00	0.00	43530.00	0.08
2877.00	186.00	104050.20	55.00	936.00	0.82
1051.00	196.00	38000.20	11.00	9581.00	0.42
3327.00	91.00	120330.40	32.00	171.00	0.67
1861.00	144.00	67323.20	49.00	0.00	0.62
1009.00	181.00	36495.50	6.00	3464.00	0.36
2520.00	48.00	91137.50	9.00	4318.00	0.42
1957.00	100.00	70799.20	30.00	26.00	0.49
1511.00	104.00	54638.40	38.00	1156.00	0.48
1680.00	73.00	60765.60	21.00	1530.00	0.39
844.00	109.00	30520.20	27.00	3415.00	0.36
1560.00	79.00	56407.10	21.00	377.00	0.38
1560.00	79.00	56407.10	21.00	0.00	0.38
1024.00	84.00	37048.90	24.00	0.00	0.33
758.00	51.00	27409.60	13.00	4650.00	0.22
0.00	0.00	0.00	0.00	14643.00	0.03
662.00	84.00	23962.60	38.00	333.00	0.35
827.00	58.00	29905.40	12.00	60.00	0.22
1368.00	20.00	49473.30	0.00	240.00	0.20
875.00	51.00	31634.30	11.00	25.00	0.22
0.00	0.00	0.00	0.00	10716.00	0.02
317.00	38.00	11473.10	21.00	3119.00	0.18
266.00	31.00	9628.50	18.00	2890.00	0.15
266.00	33.00	9639.30	11.00	2593.00	0.12
649.00	18.00	23481.60	10.00	663.00	0.15
0.00	0.00	0.00	0.00	5542.00	0.01
430.00	18.00	15567.60	10.00	0.00	0.12
222.00	35.00	8037.00	8.00	0.00	0.10
669.00	2.00	24205.00	0.00	0.00	0.09
223.00	33.00	8062.30	1.00	123.00	0.07
154.00	6.00	5562.90	10.00	1684.00	0.07
165.00	28.00	5978.90	0.00	72.00	0.05
126.80	17.00	4564.70	0.00	178.00	0.04
154.00	6.00	5562.90	10.00	23.00	0.07
73.00	16.00	2625.90	0.00	58.00	0.03
90.00	6.00	3255.30	0.00	0.00	0.02
13.00	4.00	477.40	0.00	0.00	0.01

Figure 18. Paste the deleted contents



MAN_HR	FAILURE	COST	CANN	MIC_HRS	Rank
6435.00	317.00	232764.80	68.00	7538.00	1.49
3960.00	174.00	143244.10	47.00	1166.00	0.91
2877.00	186.00	104050.20	55.00	936.00	0.82
3327.00	91.00	120330.40	32.00	171.00	0.67
1861.00	144.00	67323.20	49.00	0.00	0.62
1957.00	100.00	70799.20	30.00	26.00	0.49
1511.00	104.00	54638.40	38.00	1156.00	0.48
2520.00	48.00	91137.50	9.00	4318.00	0.42
1051.00	196.00	38000.20	11.00	9581.00	0.42
1680.00	73.00	60765.6			
1560.00	79.00	56407.1			
1560.00	79.00	56407.1			
1009.00	181.00	36495.5			
844.00	109.00	30520.2			
662.00	84.00	23962.6			
1024.00	84.00	37048.9			
827.00	58.00	29905.4			
758.00	51.00	27409.6			
875.00	51.00	31634.3			
1368.00	20.00	49473.3			
317.00	38.00	11473.1			
266.00	31.00	9628.5			
649.00	18.00	23481.6			
266.00	33.00	9639.3			
430.00	18.00	15567.6			
222.00	35.00	8037.0			
669.00	2.00	24205.0			
0.00	0.00	0.0			
154.00	6.00	5562.9			
154.00	6.00	5562.90	10.00	23.00	0.07
223.00	33.00	8062.30	1.00	123.00	0.07
165.00	28.00	5978.90	0.00	72.00	0.05
126.00	17.00	4564.70	0.00	178.00	0.04
73.00	16.00	2625.90	0.00	58.00	0.03
0.00	0.00	0.00	0.00	14643.00	0.03
0.00	0.00	0.00	0.00	10716.00	0.02
90.00	6.00	3255.30	0.00	0.00	0.02
0.00	0.00	0.00	0.00	5542.00	0.01
13.00	4.00	477.40	0.00	0.00	0.01

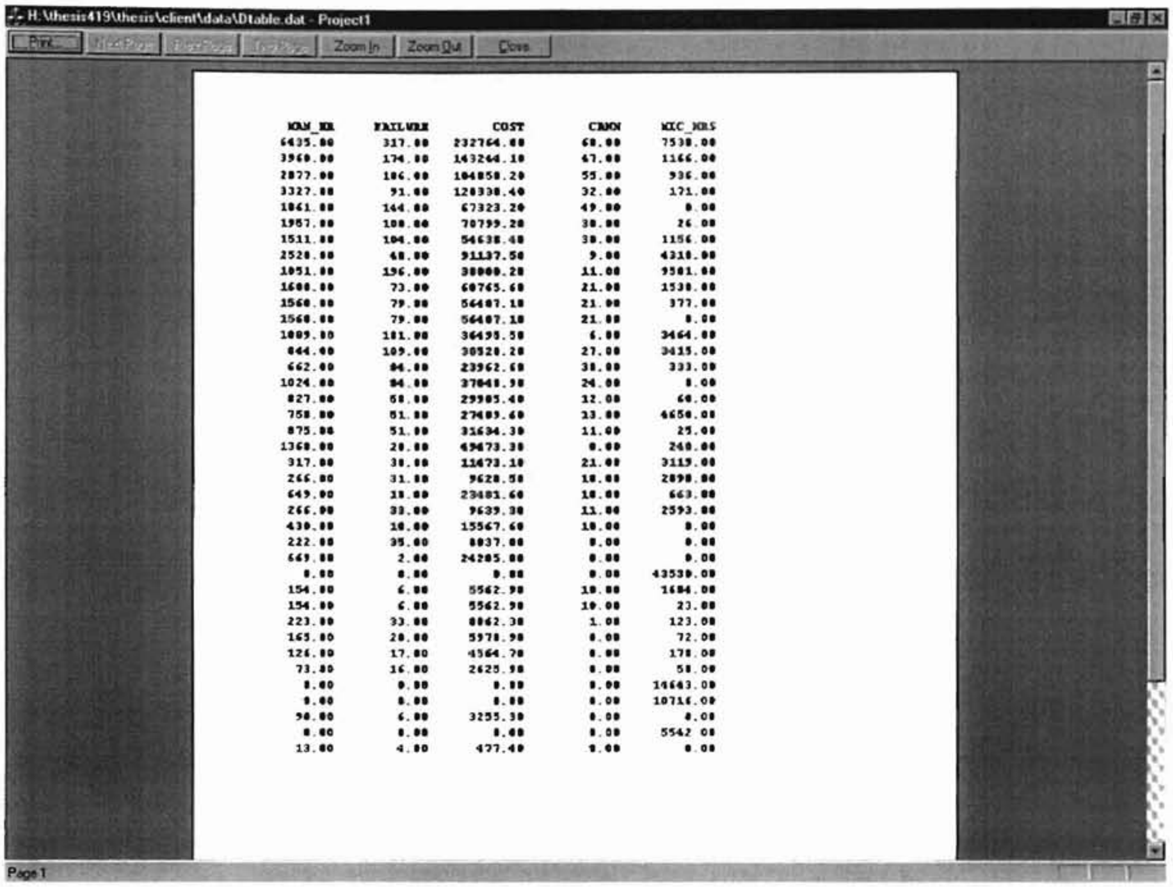
Print

Printer
Name: HP LaserJet 8000 L1
Status: Ready
Type: HP LaserJet 8000 Series PS
Where: LPT1:
Comment:
Print to file

Print range
[✓ All]
Pages from: 1 to:
Selection

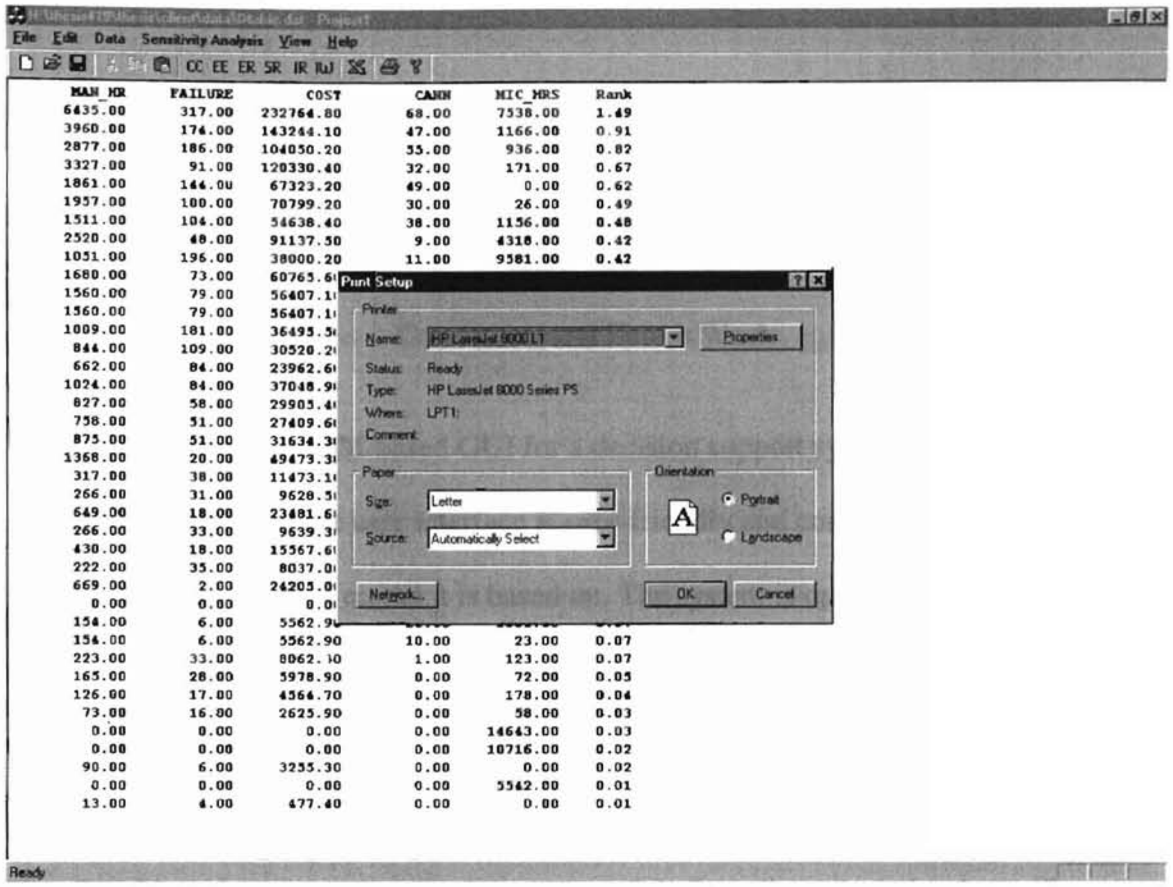
Copies
Number of copies: 1
Collate

Figure 19. Print dialog for page setup



MAN_HR	FAILURE	COST	CANN	MIC_HRS
6435.00	317.00	232764.00	68.00	7538.00
3960.00	174.00	143264.10	47.00	1166.00
2877.00	186.00	104858.20	55.00	936.00
3327.00	91.00	120338.40	32.00	171.00
1861.00	144.00	67323.20	49.00	0.00
1957.00	108.00	70799.20	38.00	26.00
1511.00	104.00	54638.40	38.00	1156.00
2528.00	48.00	91137.50	9.00	4318.00
1051.00	196.00	38000.20	11.00	9581.00
1600.00	73.00	60765.60	21.00	1538.00
1560.00	79.00	56407.10	21.00	377.00
1560.00	79.00	56407.10	21.00	0.00
1009.00	181.00	36495.50	6.00	3464.00
844.00	109.00	30520.20	27.00	3415.00
662.00	84.00	23962.60	38.00	333.00
1024.00	84.00	37048.90	24.00	0.00
827.00	68.00	29905.40	12.00	60.00
758.00	81.00	27409.60	13.00	4650.00
875.00	51.00	31634.30	11.00	25.00
1368.00	20.00	49473.30	0.00	240.00
317.00	38.00	11473.10	21.00	3119.00
266.00	31.00	9620.50	10.00	2898.00
649.00	18.00	23481.60	18.00	663.00
266.00	33.00	9639.30	11.00	2593.00
430.00	10.00	15567.60	10.00	0.00
222.00	35.00	8037.00	0.00	0.00
669.00	2.00	24205.00	0.00	0.00
0.00	0.00	0.00	0.00	43530.00
154.00	6.00	5562.90	10.00	1684.00
154.00	6.00	5562.90	10.00	23.00
223.00	33.00	8062.30	1.00	123.00
165.00	20.00	5978.90	0.00	72.00
126.00	17.00	4564.70	0.00	178.00
73.00	16.00	2625.90	0.00	58.00
0.00	0.00	0.00	0.00	14643.00
0.00	0.00	0.00	0.00	10716.00
90.00	6.00	3255.30	0.00	0.00
0.00	0.00	0.00	0.00	5542.00
13.00	4.00	477.40	0.00	0.00

Figure 20. Print preview of current view



MAN_HR	FAILURE	COST	CANN	MIC_HRS	Rank
6435.00	317.00	232764.80	68.00	7538.00	1.49
3960.00	174.00	143244.10	47.00	1166.00	0.91
2877.00	186.00	104050.20	55.00	936.00	0.82
3327.00	91.00	120330.40	32.00	171.00	0.67
1861.00	144.00	67323.20	49.00	0.00	0.62
1957.00	100.00	70799.20	30.00	26.00	0.49
1511.00	104.00	54638.40	38.00	1156.00	0.48
2520.00	48.00	91137.50	9.00	4318.00	0.42
1051.00	196.00	38000.20	11.00	9581.00	0.42
1680.00	73.00	60765.6			
1560.00	79.00	56407.1			
1560.00	79.00	56407.1			
1009.00	181.00	36495.5			
844.00	109.00	30520.2			
662.00	84.00	23962.6			
1024.00	84.00	37048.9			
827.00	58.00	29905.4			
758.00	51.00	27409.6			
875.00	51.00	31634.3			
1368.00	20.00	49473.3			
317.00	38.00	11473.1			
266.00	31.00	9628.5			
649.00	18.00	23481.6			
266.00	33.00	9639.3			
430.00	18.00	15567.6			
222.00	35.00	8037.0			
669.00	2.00	24205.0			
0.00	0.00	0.0			
154.00	6.00	5562.9			
154.00	6.00	5562.90	10.00	23.00	0.07
223.00	33.00	8062.30	1.00	123.00	0.07
165.00	28.00	5978.90	0.00	72.00	0.05
126.00	17.00	4566.70	0.00	178.00	0.04
73.00	16.00	2625.90	0.00	58.00	0.03
0.00	0.00	0.00	0.00	14643.00	0.03
0.00	0.00	0.00	0.00	10716.00	0.02
90.00	6.00	3255.30	0.00	0.00	0.02
0.00	0.00	0.00	0.00	5542.00	0.01
13.00	4.00	477.40	0.00	0.00	0.01

Print Setup

Printer

Name: HP LaserJet 8000 L1

Status: Ready

Type: HP LaserJet 8000 Series PS

Where: LPT1:

Comment:

Paper

Size: Letter

Source: Automatically Select

Orientation

[✓ Portrait] Landscape

Figure 21. Print setup for paper size and orientation

CHAPTER V

Conclusion and Future Work

This thesis provides a COM-based GUI for a decision support system. Implemented in Microsoft Visual C++, the user interface is user-friendly and convenient to update in the future thanks to the COM model it is based on. The system is open to variety of input and output data sources. It also will be able to exchange data with Microsoft Excel and gives user more choices when exporting data to other applications.

This COM-based system separates the user interface from the calculation logic. It successfully establishes the communication between these two parts based on COM framework. Although it is possible to distribute the system on the network, developers find it hard to configure the connected computers for security reasons. Also the firewalls that are commonly used to protect Intranet restrict the establishment of such communication. Microsoft has been developing a new standard called SOAP (Simple Object Access Protocol) that is using XML (Extensible Stylesheet Language) as network communication media. SOAP applications can be easily deployed on the network since it uses port 80 (standard http port) to communicate without interference of firewalls [16]. Further research in this area is needed to enhance the practical use of the distributed component systems.

REFERENCES

1. Cockbum, A. A. R. The impact of Object-Orientation on Application Development, IBM Systems Journal, Vol. 38, 308-332, 1998.

2. Collins, D. Designing Object-Oriented User Interfaces, Benjamin/Cummings, 1995.

3. Corry, C., Mayfield, V., Cadman, J., Morin, R. COM/DCOM Primer Plus, Sams Publishing, 1999.

4. Eddon, G., Eddon, H., Inside Distributed COM, Microsoft Press, 1998.

5. Fabian, R. The GUI Challenge, Info Canada, Vol. 16, 7-10, September, 1991.

6. George, K. M. Computer Method for Sustainability Ranking, AFOSR report, 1996.

7. Hix, D. and Hartson, R. H. Developing User Interface: Ensuring Usability Through Product and Process, John Wiley & Sons, 1993.

8. Jewell, D. Polishing Windows, Addison-Wesley Publishing Company, 1994.

9. Johnson, R. A., and Wichern, D. W. Applied Multivariate Statistical Analysis, Prentice-Hall, Inc., 1982.

10. Laurel, B., Ed. The Art of Human-Computer Interface Design, Addison-Wesley, 1991.

11. Lee, G. Object-Oriented GUI Application Development, PTR Prentice Hall, 1993.

12. Lewandowski, M. S. Frameworks for Component-based Client/Server Computing, ACM Computing Surveys, Vol. 30, 3-27, March 1998.

13. Lionel, C. B., Morasca, S., Basili, R. V. Defining and Validating Measures for Object-Based High-Level Design, IEEE Transactions on Software Engineering, Vol. 25, 722-743, September/October, 1999.

14. Marcus, A. Graphic Design for Electronic Documents and User Interfaces, ACM Press, 1992.

15. Mayhew, D. Principles and Guidelines in Software User Interface Design, PTR Prentice-Hall, Inc., 1992.

16. Microsoft Corp., Digital Equipment Corp., The Component Object Model Specification, http://msdn.microsoft.com/isapi/msdnlib.idc?theURL=/library/specs/s1d139.htm, 1995

17. Thimbleby, H. User Interface Design, Addison-Wesley Publishing Company, 1990.

18. Treu, S. User Interface Design A Structured Approach, Plenum Press, 1994.

19. Wen, S. Application of Principal Component Analysis to Decision Support System, MS Thesis, Oklahoma State University, 1997.

20. Wood, L. E. User Interface Design Bridging the Gap from User Requirements to Design, CRC Press LLC, 1998.

21. Zaratian, B. Microsoft Visual C++ 6.0 Programmer's Guide, Microsoft Press, 1998.

22. Zave, P. A Component-Based approach to Telecommunications Software, IEEE Software, Vol. 15, 70-78, September, 1998.

23. Zetie, C. Practical User Interface Design, McGraw-Hill, 1995.

APPENDEX A

ACRONYMS

API	Application Programming Interface
ATL	Active Type Library
COM	Component Object Model
DCE	Distributed Computing Environment
GUI	Graphical User Interface
GUID	Globally Unique Identifier
IDE	Integrated Development Environment
IDL	Interface Definition Language
JDK	Java Development Kit
OSF	Open Software Foundation
PCA	Principal Component Analysis
SDK	Standard Development Kit
SOAP	Simple Object Access Protocol
UML	Unified Model Language
URL	Universal Resource Locator
XML	Extensible Stylesheet Language

VITA

Yu An

Candidate for the Degree of

Master of Science

Thesis: A COM-BASED GRAPHICAL USER INTERFACE FOR A DECISION SUPPORT SOFTWARE

Major Field: Computer Science

Biographical:

Personal Data: Born in Chong Qing, China, September, 1973, the youngest child of Wu Chaozheng and Ding Zhengguo.

Education: Graduated from Nankai High School, Chong Qing, China in June, 1991; received Bachelor of Science degree in Biochemistry, minor in Applied Computer Science from Sichuan University in July 1995. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in May, 2000.

Experience: Research Assistant from May, 1999 to December, 1999 in Plant and Soil Sciences Department at Oklahoma State University.

Professional Memberships: ACM (Association of Computing Machinery)